Università di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Formal Modeling and Simulation of Biological Systems with Delays

Giulio Caravagna

Supervisor
Roberto Barbuti

Supervisor
Paolo Milazzo

June, 2011

Abstract

Delays in biological systems may be used to model events for which the underlying dynamics cannot be precisely observed, or to provide abstraction of some behavior of the system resulting in more compact models. Deterministic modeling of biological systems with delays is usually based on Delay Differential Equations (DDEs), an extension of ordinary ones where the derivative of the unknown function depends on past-states of the system. Stochastic modeling is done by using non-Markovian stochastic processes, namely processes whose sojourn time in a state and the probability of a transition are not necessarily exponentially distributed. Moreover, in the literature Delay Stochastic Simulation Algorithms (DSSAs) have been proposed as extension of a well-known Stochastic Simulation Algorithm (SSA) for non-delayed models.

In the first part of the thesis we study different DSSAs. The first two algorithms we present treat delays by means of some scheduling policy. One algorithm is based on the idea of "delays as durations" approach (DDA), the other is based on a "purely delayed" interpretation (PDA) of delays. We perform deterministic and stochastic analysis of a cell cycle model. Results suggest that the algorithms differ in the sense that the PDA, even in a naive definition, is more suitable than the DDA to model systems in which species involved in a delayed interaction can be involved at the same time in other interactions. We investigate the mathematical foundations of the algorithms by means of the associated Delay Chemical Master Equations (DCMEs). Result of the comparison is that both are correct with respect to their DCMEs, but they refer to systems ruled by different DCMEs. These results endorse our intuition on the difference between the algorithms. Improvements of the naive PDA lead to a definition of a more precise algorithm, the PDA with markings, which is finally combined with the DDA.

The last algorithm we propose, the DSSA with Delayed Propensity Functions (DPF), is inspired by DDEs in its definition. In the DPF changes in the current state of the system are affected by the history of the system. We prove this algorithm to be correct, we show that systems ruled by the DCME associated with this algorithm are the same ruled by those target of the PDA, and then argue that the DPF and the PDA are equivalent.

We then study formal models of biological systems with delays. Initially, we start adding delayed actions to CCS. This leads to two new algebras, CCSD and CCSP, the former where actions follow the DDA, the latter where actions follow the PDA. For both the algebras we define a notion of process, process configurations, and we give a Structural Operational Semantics (SOS) in the Starting-Terminating (ST) style, meaning that the start and the completion of an action are observed as two separate events, as required by actions with delays. We define bisimulations in both CCSD and CCSP, and we prove bisimulations to be congruences even in the case of delayed actions.

Finally, we enrich the stochastic process algebra BIO-PEPA with the possibility of assigning DDA-like delays to actions, yielding a new non-Markovian stochastic process algebra: BIO-PEPAD. This is a conservative extension meaning that the original syntax of BIO-PEPA is retained and moving from existing BIO-PEPA models to models with delays is straightforward. We define process configurations and a SOS in the ST-style for BIO-PEPAD. We formally define the encoding of BIO-PEPAD models in Generalized semi-Markov Processes (GSMPs), a class of non-Markov processes, in input for the DDA and in sets of DDEs. Finally, we investigate the relation between BIO-PEPA and BIO-PEPAD models.

Contents

List of Figures

List of Algorithms

List of Acronyms

SSA	*Stochastic Simulation Algorithm*
DSSA	*Delay Stochastic Simulation Algorithm*
DDA	*Delay-as-Duration Approach*
PDA	*Purely Delayed Approach*
MPDA	*Purely Delayed Approach with Markings*
DPF	*Delayed Propensity Functions*
FULL DSSA	*Full Delay Stochastic Simulation Algorithm*
ODE	*Ordinary Differential Equation*
DDE	*Delay Differential Equation*
CME	*Chemical Master Equation*
DCME	*Delay Chemical Master Equation*
CTMC	*Continuous-Time Markov Chain*
GSMP	*Generalized Semi-Markov Process*
LTS	*Labelled Transition System*
SOS	*Structural Operational Semantics*
ST	*Starting-Terminating*
CCS	*Calculus of Concurrent Systems*
CCSP	*Calculus of Concurrent Systems with Delay-as-Duration Approach*
CCSD	*Calculus of Concurrent Systems with Purely Delayed Approach*
BIO-PEPA	BIO-PEPA
BIO-PEPAD	BIO-PEPA *with Delays*

Chapter 1

Introduction

In this chapter we provide motivations for this thesis. We start contextualizing this work in the interdisciplinary field of research named *Systems Biology*. We then briefly describe the contribution to the field and the structure of the thesis.

1.1 Motivations

Systems Biology in brief. Biochemistry, often conveniently described as the study of the chemistry of life, is a multifaceted science that includes the study of all forms of life and that utilizes basic concepts derived from Biology, Chemistry, Physics and Mathematics to achieve its goals. Biochemical research, which arose in the last century with the isolation and chemical characterization of organic compounds occurring in nature, is today an integral component of most modern biological research.

Most biological phenomena of concern to biochemists occur within small, living cells. In addition to understanding the chemical structure and function of the biomolecules that can be found in cells, it is equally important to comprehend the organizational structure and function of the membrane-limited aqueous environments called cells. Attempts to do the latter are now more common than in previous decades. Where biochemical processes take place in a cell and how these systems function in a coordinated manner are vital aspects of life that cannot be ignored in a meaningful study of biochemistry. Cell biology, the study of the morphological and functional organization of cells, is now an established field in biochemical research.

Computer Science and Mathematics can help the research in cell biology in several ways. For instance, it can provide biologists with models and formalisms able to describe and analyze complex systems such as cells. This rather new interdisciplinary field of research is named *Systems Biology* (Kitano, 2001; Ideker *et al.*, 2001; Sauer *et al.*, 2007).

Modeling biological systems. The approach which is used to solve a problem of system biology is typically the following: firstly the biological system has to be identified in all of its components, if possible. In particular all the involved elements and the interesting events have to be identified; this is generally one of the major problems because information may be lacking since some natural phenomena can not be properly observed. This results in a partial knowledge of the biological system itself. Such considerations are completely general and independent on the level of abstraction of the target systems, indeed this applies when we refer to molecules and reactions, or to individuals and events within a population. The general spectrum of systems biology is wide and consider biological systems at any level of abstraction.

Whenever the system has been identified, it is possible to build models solely *deterministic*, *stochastic*, or even combinations of both. Models are built on the data which has been carried out by observing real natural systems.

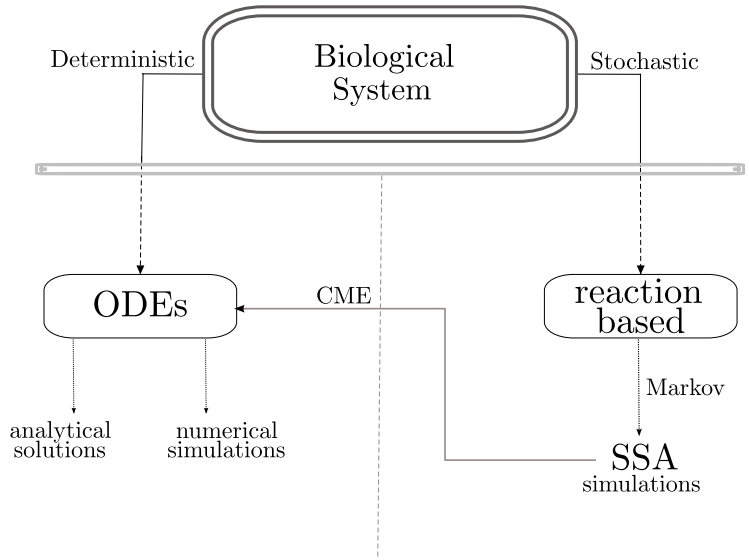

FIGURE 1.1: The modeling schema in Systems Biology.

A deterministic model is generally a system of *continuous* differential equations which models the variation of concentrations of the involved components and whose terms are the modeling of the observed events. In the simplest case equations are *Ordinary Differential Equations* (ODEs), and only in a few cases can be solved analytically finding the solution of the equation, the equilibria and the bifurcation points. However, numerical simulations are always possible. Results obtained analyzing these models are satisfactory when modeling systems involving a huge number of components. Differently, when the size of the biological systems is small, deterministic models may not show some dynamics which are instead observed in real systems. This is caused by the fact that differential equations represent discrete quantities with continuous variables, and when quantities are close to zero this may become a too imprecise approximation. To overcome this incompleteness, stochastic models can be defined.

A stochastic model can be defined via the same principles used to derive a deterministic one. Typically, such a model is specified in a reaction-style notation. Such a model can potentially exhibit behaviors not captured by the deterministic counterpart since the involved quantities vary *discretely* and *stochastically*. Mathematically, such models are typically stochastic processes which, in most cases, are Markov since this permits easier analysis. To analyze Markov models *Stochastic Simulation Algorithms* (SSAs) (Gillespie, 1976) have been defined to compute a single time evolution of the modeled system. One of the most interesting features of these algorithms is the introduction of the notion of *propensity function* for a reaction. Such functions are used to compute the probability of reactions to happen in a medium, once molecules are randomly distributed in space. The input of these algorithms are systems described as a set of reactions (i.e. rewriting rules) and an initial state. The time evolution is given by the probabilistic sequence of firing of the reactions in the system state. Such algorithms are logically related to deterministic systems via *Chemical Master Equations* (CMEs) (Gillespie, 1976) associated to simulated systems, these equations describes the time evolution of the probability of the system to occupy each one of a discrete set of states.

In FIGURE 1.1 the modeling schema we refer to is represented.

The use of delays. As we said, a crucial problem in modeling real systems is the identification of the components and events involved. Here difficulties come from the inherent experimental approach to this science. Delays can appear in a biological system at any level of abstraction. We go through this consideration by informally discussing two very simple scenarios. Firstly, let

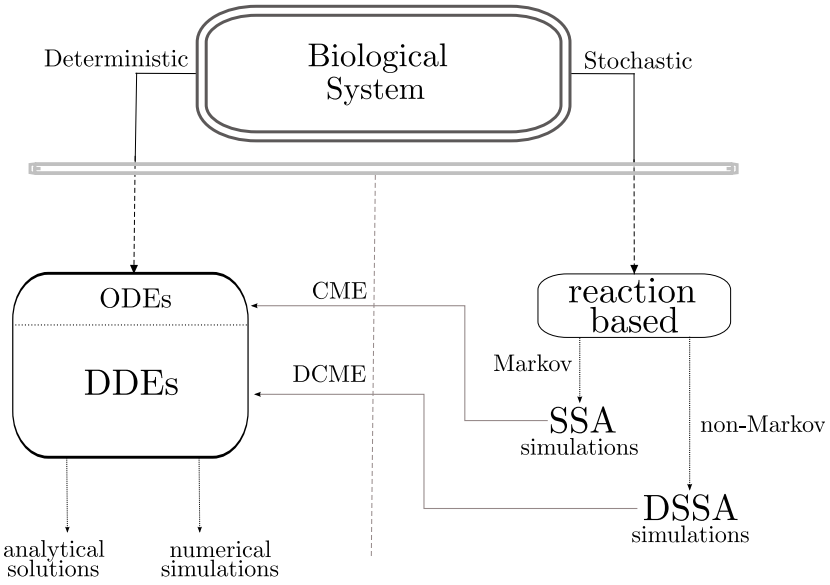

FIGURE 1.2: The modeling schema extended with delays.

us consider a complex dynamics (*macro-event*) decomposed in a series of sequential sub-events (*micro-events*): to explicitly model such dynamics we must have full quantitative information about all the sequential sub-events. This requirement implies that, if some information is missing then a full model cannot be described, and this is quite a common scenario in real modeling. If this is the case, we can think about a raw abstraction of this system by considering, instead of the micro-events, the single-step macro-event, assuming that we have enough information for it to be modeled. Delays come into play at this stage when the average time for completion of the macro-event is known. Indeed, such an information can be used to have a more precise model, as we shall see in the rest of the thesis. Of course even though this is an *abstraction* of the exact model of the micro-events, this turns out to be the best that we can do in some situations. As we said, there is another scenario in which delays turn out to be useful. Namely, when a system is too complex to analyze, then using a similar assumption to above, we can replace a collection of micro-events by a model with delay at the macro-event level. In this case, the delay is used as a *model-reduction* technique which makes the model smaller, and the analysis potentially feasible.

Models with delays. Deterministic modeling of biological systems with delays is mainly based on *Delay Differential Equations* (DDEs) obtained by generalizing ODEs. These equations are generally harder than ODEs to be analyzed. Also, as for ODEs the analysis of DDEs can become imprecise due to the approximation introduced by representing discrete quantities by continuous variables when quantities are close to zero. Thus techniques for performing stochastic analysis of systems with delays have also been developed. Stochastic models with delays result in non-Markovian stochastic processes. Starting from the SSAs for some of such processes *Delay Stochastic Simulation Algorithms* (DSSAs) have been defined (Bratsun *et al.*, 2005; Barrio *et al.*, 2006; Barbuti *et al.*, 2009b). In these cases such DSSAs are logically related to deterministic systems via some *Delayed Chemical Master Equation* (DCMEs) (Barrio *et al.*, 2006), an extension with delays of the CMEs.

In FIGURE 1.2 is represented the modeling schema extended with delays.

Formalisms from Computer Science. Experimental studies in biology have been producing a lot of data in the last years through enhanced techniques for analyzing biological systems; this increased in size and complexity existing models. Consequently, new methods supporting devel-

opment of complex models are required. Over the past decade Computer Science has been helpful in defining formalisms to describe complex systems. *Concurrent systems* theory is at the basis of most of the formalisms applied in systems biology.

There exist many formal languages, either based on *process algebras*, term–rewriting systems or different mathematical structures, worth noting: the Calculus of Concurrent Systems (CCS) (Milner, 1980), the π-CALCULUS (Milner *et al.*, 1992; Regev *et al.*, 2001; Palamidessi, 2003) with its stochastic (Priami, 1995; Cardelli *et al.*, 2009) and continuous (Kwiatkowski & Stark, 2008) variants, BETA BINDERS (Priami & Quaglia, 2005; Guerriero *et al.*, 2007), BLENX (Dematté *et al.*, 2008), BIOAMBIENTS (Regev *et al.*, 2001), BRANE CALCULI (Cardelli, 2005), PEPA (Hillston, 1996; Calder *et al.*, 2006), BIO-PEPA (Ciocchetta & Hillston, 2009; Ciocchetta & Hillston, 2008; Akman *et al.*, 2009), PETRI NETS (Murata, 1989; Reddy *et al.*, 1993) and some of their variants (Jensen, 1992; Leea *et al.*, 2006), LBS (Pedersen & Plotkin, 2010), κ (Danos *et al.*, 2009a; Danos *et al.*, 2009b), the CALCULUS OF LOOPING SEQUENCES (Milazzo, 2007; Barbuti *et al.*, 2008a; Barbuti *et al.*, 2008b) and BIGRAPHS (Damgaard *et al.*, 2008a; Damgaard *et al.*, 2008b; Krivine *et al.*, 2008). Some of these formalisms such as CCS, the π-CALCULUS or PEPA have been defined to model different concurrent systems; however, subsequently they have been successfully applied in Systems Biology. As often happens, this motivated in defining ad-hoc languages such as BLENX, BIO-PEPA or κ. Also, biology inspired new models of computations so, for instance, DNA computing (Păun *et al.*, 1998) or P-SYSTEMS (Păun, 2002) have been defined to compute by using cells or other types of molecules. In particular, the connection between P-SYSTEMS and process algebras has been actively studied (Cardelli & Păun, 2006; Barbuti *et al.*, 2008c; Barbuti *et al.*, 2008d; Barbuti *et al.*, 2010c; Barbuti *et al.*, 2010d).

A desired feature of a formalism is being *compositional*. A formal system is compositional if the semantics of composite object can be determined from the semantics of the components. Compositional systems therefore can be analyzed in a modular way, non compositional systems must be analyzed in their whole. Differential equations are not compositional since the solution of a set of equations can not be determined as a combination of the individual solutions of the equations. Differently, process algebras are typically compositional because of the semantics of their cooperation operators. In this approach chemical reacting entities are described by processes and biological reactions are modeled as cooperation/synchronization between processes, the act of synchronizing of communicating processes is interpreted as the firing of a reaction. In this thesis, for all the reasons we outlined we concentrate on process algebras when considering formal languages. However, all the theories we develop in the context of process algebras could be equivalently defined in different contexts (i.e. rewriting systems).

1.2 Contributions

In this thesis we address two major issues: firstly *stochastic simulation* and secondly *formal modeling* of biological systems with delays.

In the first part of the thesis we focus on DSSAs. We analyze a well-known algorithm (Barrio *et al.*, 2006) and its interpretation of delays, which is named delay-as-durations (DDA). We argue that such an algorithm is not suitable to simulate systems in which species involved in a delayed interaction can be involved at the same time in other interactions. Then we define a novel DSSA based on an pure interpretation of delays (PDA) which behaves more properly, even in its naïve definition, in simulating such systems. We provide both experimental evidences of the difference between the DDA and the PDA and we analyze mathematical foundations of both the algorithms. The PDA is then improved to avoid some inaccurate behaviors which may happen in its naïve definition. This new PDA, named PDA with markings, is then combined with the DDA.

Both these algorithms are based on the idea of delays as a scheduling policy, where the policy depends on the delays interpretation. For the sake of investigating further interpretations of delays we define a DDEs-inspired DSSA where delays represent dependancies on past states of the simulated system. Investigating the mathematical foundations of this algorithm we prove it to be equivalent to the PDA.

This concludes the first part in which we rigorously present five algorithms for simulating stochastic systems with delays, and we prove relations among them. In the second part of the thesis we address the problem of supporting actions with delays in formal languages. To this extent, we provide two major results.

The former regards extending CCS to support non-instantaneous actions. More precisely, we define two extensions of CCS where actions follow either the delay-as-duration (CCSD) or the purely delayed approach (CCSP). Both the languages are conservative extensions of CCS, in the sense that the syntax of CCS processes is retained. Both languages are able to describe processes in which non-instantaneous actions are started and not completed. This introduces, in the case of CCSP, notions of competing actions internal to processes. For both the languages we define a Structural Operational Semantics (SOS) (Plotkin, 1981; Plotkin, 2004) in the Starting-Terminating (ST) style (van Glabbeek & Vaandrager, 1987; Bravetti *et al.*, 1998) and bisimulation relations (Milner, 1980; Park, 1981). We discuss the effect of non-instantaneous actions on classical CCS operators, and we prove bisimulations to be congruences in both CCSD and CCSP.

The latter result of the second part concerns the enrichment of the stochastic process algebra Bio-PEPA with the possibility of assigning delays to actions following the delay-as-duration approach, yielding a new non-Markovian stochastic process algebra: Bio-PEPAD. Bio-PEPAD processes are defined with the original syntax of Bio-PEPA. We define notions similar to the one introduced for CCSD, namely process configurations, we define systems and a SOS in the ST-style for Bio-PEPAD. We formally define the encoding of Bio-PEPAD models in Generalized semi-Markov Processes (GSMPs), a class of non-Markov processes, in input for the DDA and in sets of DDEs. We prove results stating the relation between Bio-PEPA and Bio-PEPAD models.

1.3 Related Work

Stochastic simulation of non-delayed systems is mostly done by the SSA (Gillespie, 1976; Gillespie, 1977) and other SSA-based algorithms. Such an algorithm became more used in the last decade, whereas it has been defined in late 70's. The theory of stochastic simulation of delayed system is, instead, a much newer research topic.

One of the first works on DSSAs is (Barrio *et al.*, 2006). In there, possibly different algorithms are informally discussed. For the informally discussed algorithms no clear mathematical foundations are investigated. For a specific variant of this algorithm more considerations are given, even though correctness of that algorithm is not discussed. Such a variant is based on the interpretation of delays which in (Barbuti *et al.*, 2009b) has been termed to be the delay-as-durations approach. Nowadays that algorithm, hereby named DDA, represents the most used simulation algorithm for systems with delays. In (Roussel & Zhu, 2006; Leier *et al.*, 2007) the DDA is successfully applied to the simulation of a model of gene transcription and translation; we discuss those kind of models in Chapter 3. The DDA is based on the idea of scheduling reactions once that the reactants consumed by the reaction firing have been removed by the simulation state. As for all the scheduling-based algorithm, policies for handling scheduled reactions are defined based on rejection of generated random numbers. Such policies perform state-changes when scheduled reactions are handled. Summarizing, in this algorithm for each reaction two detached state-changes are induced. In (Barrio *et al.*, 2006) a DCME is presented and claimed to be the one related to systems simulated by the DDA. However, accordingly to our results, such a DCME is not related to such systems since it refers to systems in which a single state-change is induced by a single reaction. We discuss this algorithm and our results in Chapter 4 and in Chapter 5.

Systems where is correct to think about a unique state-change induced by a reaction are those simulated by the algorithm introduced in (Bratsun *et al.*, 2005). Such an algorithm is the first which implicitly adopted the approach we discuss in Chapter 5. However, mathematical foundations of the the algorithm presented in (Bratsun *et al.*, 2005) are not investigated. We present an algorithm based on a purely delayed approach in the use of delays in Chapter 5 and for that algorithm we discuss its mathematical foundations.

Some other DDA-based DSSAs have been defined in the last years. So, for instance, in (Cai,

2007) complex data structures are used so that rejection of generated random numbers is avoided. Moreover, mathematical foundations of the algorithms presented in (Cai, 2007) are studied and the correctness of the algorithm is discussed. Such a new algorithm is harder to be coded than the original DDA, but it is more time-efficient in practical usage. Similar results are obtained in (Anderson, 2007) where the algorithm presented in (Cai, 2007) is further improved by removing the complex data structures introduced to decrease the number of rejected random values. In (Zhou *et al.*, 2008) these DSSAs are analyzed by means of the moment probability functions instead, as in their corresponding works, by means of the probability density functions.

In (Anderson, 2007) for the first time is recognized a connection between some of the DSSAs and a class of underlying non-Markovian stochastic processes. Moreover, in (Schlicht & Winkler, 2008) a a constructive proof of the existence of the non-Markovian stochastic process and a derivation of the involved probabilities is given. In (Jansen, 1995; Anderson, 2007; Shahrezaei *et al.*, 2008) are also studied generalized SSA-based algorithms with time-dependent propensity functions. For the sake of studying DSSAs a result presented in (Shahrezaei *et al.*, 2008) is recalled in CHAPTER 2.

The DSSAs we discussed up to now are *exact*. Sometimes when thinking about DSSAs it is convenient to decrease the precision of the algorithm and improve simulation performances. In this sense, a similar work has been done in non-delayed systems by proposing approximations of exact SSAs (Gillespie, 2001; Rathinam *et al.*, 2003; Cao *et al.*, 2005). In (Tian *et al.*, 2007) approximations are obtained by describing DDA-based systems by means of differential equations embedding stochastic and delayed effects. Such kind of equations find their conceptual base either in the *Fokker-Planck* or in the *Langevin* equations, as it happens for those presented in (Cao *et al.*, 2005).

More complex form of delays have been studied for instance in (Marquez-Lago *et al.*, 2010; Zhu *et al.*, 2007; Ribeiro *et al.*, 2009). In (Marquez-Lago *et al.*, 2010) such a work spatial information of systems are introduced by means of ad-hoc probability distributed time delays. Such distributions are obtained by specialized experiments on the target system and then used in combination with the DSSAs of (Barrio *et al.*, 2006). Differently, in (Zhu *et al.*, 2007; Ribeiro *et al.*, 2009) reactions with multiple delays are considered.

As far as formal methods is concerned, process algebras has been studied in a lot of different formats: some including time, some including stochastic features and some combining both these aspects. So, for instance, process algebras with discrete or continuous notions of time has been defined (Moller & Tofts, 1990; Baeten & Bergstra, 1991; Nicollin & Sifakis, 1994; Hennessy & Regan, 1995) so that quantitative timing aspects can be investigated. In the context of Systems Biology, we are required to have a continuous-time domain underlying our systems.

In some semantics the notion of time is explicit, in the sense that a state of a system is given by some process and a global clock. The system can perform transitions modeling some changes in either the state process, once actions are performed, or in the global clock, by consuming time. In other cases, it is convenient to describe passage of time relative to previous actions. In the first case time is *absolute* (Corradini, 2000), whereas in the second is *relative* (Baeten & Bergstra, 1991). In (Baeten & Bergstra, 1997) the combinations of both the approaches is discussed, yielding to a notion of *parametric* timing. In most of these algebras explicit operators to spend time are required.

There are cases in which the notion of time is substituted by other features such as stochastic aspects of actions (Priami, 1995; Bravetti & Gorrieri, 2002; Ciocchetta & Hillston, 2009). In some stochastic process algebras it turns out that actions have no duration, making them instantaneous (Priami, 1995; Ciocchetta & Hillston, 2009). In this case, state transitions model instants between two distinct actions. If timing aspects of actions are ruled by special probability distributions then they can be abstracted from the mathematical structure underlying the algebras. In this case, the global clock can be retrieved by the frequency at which actions are performed, by means of the stochastic distributions ruling actions. Actually, this kind of languages are the most used in Systems Biology since their underlying mathematical structure is particularly convenient to perform model analysis. For the sake of our purposes most of these concepts are recalled in CHAPTER 2.

When actions have a duration or when delay is possible, classical operators of algebras such as the *choice* operator, may be affected in their original interpretation (Milner, 1980). This required

authors to define notions of choice as in (Baeten & Bergstra, 1991; Nicollin & Sifakis, 1994). When stochastic aspects are considered together with durations as in (Bravetti & Gorrieri, 2002), the resulting mathematical structure is of particular interest for modeling systems with delays, as we argue in CHAPTER 2.

In (Bravetti & Gorrieri, 2002) no explicit notion of time is present in the semantics, and hence no notion of quantitative duration is possible. This is a compromise between giving semantics without explicit notion of time and modeling durational actions. One of the best thing that we can do in this case is to simply characterize an action as a sequence of detached initiation and completion. In this sense, even if we are not able to precisely model a duration in a timed-framework, we are able to recognize a non-instantaneous event in our systems. This is the approach we generally adopt and, as in (Bravetti *et al.*, 1998; Bravetti & Gorrieri, 2002), we use the Starting-Terminating (ST) style (van Glabbeek & Vaandrager, 1987; Hennessy, 1988; van Glabbeek, 1990) of non-timed semantics which turns out to be suitable to model systems with non-instantaneous actions. A similar work has been done in (Barbuti *et al.*, 2010b; Barbuti *et al.*, 2011b) where a variant of the CCS process algebra is extended to allow multiscale models of biological systems, namely models in which actions can happen at a different level of abstraction requiring different time scales.

We remark that we decided to use process algebras as the class of target formal methods since their compositional properties are desirable in modeling complex systems. However, other types of formal languages could have been used. Among all, it is worth citing TIMED AUTOMATA based on finitely many real-valued clocks (Alur & Dill, 1994) and TIMED PETRI NETS (Reisig, 1985). These languages have been enriched with features for modeling real-time systems such as probability and stochasticity. In particular STOCHASTIC PETRI NETS (Marsan, 1989), PROBABILISTIC TIMED PETRI NETS (Escalante & Dimopoulos, 1994), PROBABILISTIC TIMED AUTOMATA (Alur *et al.*, 1992) and STOCHASTIC TIMED AUTOMATA (Cassandras & Lafortune, 2007) have been defined. It is common to use basic PETRI NETS to qualitatively model chemical reacting systems without delays. In this case, to each places in the net a species is associated, and each token in a place corresponds to a molecule of the corresponding species. Transitions moving tokens from a place to another correspond to reactions.

When modeling biological systems with delays, more complex variants of basic nets such as INTERVAL TIMED COLORED PETRI NETS (van der Aalst, 1993) may be considered. In such nets time-stamps are attached to tokens to indicate the time in which they become available, in this sense this provides a way to track the time since a token is in the system. Such a mechanism of marking tokens is of inspiration for one of the algorithms we discuss in CHAPTER 7. Finally, there exist PETRI NETS models with fixed delays or stochastic delays such as those in (Ramchandani, 1973; Sifakis, 1977; Zuberek, 1980).

1.4 Structure of the Thesis

The thesis is structured as follows.

- In CHAPTER 2 we recall some background notions of probability theory, stochastic processes, differential equations, stochastic simulation of biological systems and formal languages assumed in the rest of the thesis. Particular detail is used to introduce important results such as the relation between the Stochastic Simulation Algorithm and its associated Chemical Master Equation.

- In CHAPTER 3 we discuss some target biological systems with delays. We concentrate on some basic cellular models, epidemics models and evolutionary models and we define them in both a deterministic and a stochastic fashion. For all of those, we discuss role of delays in their modeling. Finally, we present a model of the cell cycle with a delay used as a running example along the thesis. Deterministic simulations of such a model are discussed.

For the sake of clarity, the thesis is divided in two major parts, where the first is functional to the definition of the second. In the first part, from CHAPTER 4 to 7, Delay Stochastic Simulation Algorithms are presented.

- In Chapter 4 a well-known algorithm based on a notion of "delays as durations" approach (DDa), is presented. Such an algorithm is based on a scheduling policy preventing that species involved in a delayed interaction can be involved at the same time in other interactions. We discuss the mathematical foundations of the DDa by defining a Delay Chemical Master Equation (Dcme) and we prove the correctness of the DDa. We then apply the algorithm to the simulation of the cell cycle model presented in Chapter 3, and we compare results of the deterministic and stochastic simulations.

- In Chapter 5 we present a novel algorithm based on a "purely delayed" interpretation (PDa) of delays. Such an algorithm is defined so that species involved in a delayed interaction can be involved at the same time in other interactions. We apply a naïve definition of the algorithm to the simulation of the cell cycle model, and we compare results of all the simulations performed. Results suggest that this new approach is a better candidate than the DDa for such type of systems. We then investigate the mathematical foundation of the PDa. We prove the algorithm to be correct and we define a Dcme for systems simulated by such an algorithm. As expected, such a Dcme differs from the one of the DDa.

- In Chapter 6 we present a novel history-dependent algorithm (Dpf). Such an algorithm is inspired in its definition by deterministic models with delays, hence delays are used to define dependancies on the past-states of the system. As for the other algorithms we analyze its mathematical foundations. We prove this algorithm to be correct, and we show that target systems of the Dpf underly the same Dcme of PDa systems, leading to the equivalence of the two algorithms.

- In Chapter 7 we improve the naïve PDa presented in Chapter 5 by means of a notion of marking of the molecules in the system state, yielding to the definition of a new algorithm, the mPDa. We then combine this new algorithm with the DDa introduced in Chapter 4, yielding to the definition of a new algorithm able to correctly simulate systems in which for some reactions the species involved in a delayed interaction can not be involved at the same time in other interactions, whereas others can.

In the second part, we discuss formal modeling biological systems with delays.

- In Chapter 8 we extend the Calculus of Concurrent Systems (CCS) with non-instantaneous actions. More precisely, we define two extensions of CCS where actions follow either the delay-as-duration (CCSd) or the purely delayed approach (CCSp). Both the languages assume the same syntax of CCS processes, and for both we define a notion of process configuration necessary to have processes in which non-instantaneous actions are started and not completed. For both the languages we define a labeled semantics and bisimulation relations. We discuss the effect of non-instantaneous actions on classical CCS operators, and we prove bisimulations to be congruences even in the case of non-instantaneous actions.

- In Chapter 9 we enrich the stochastic process algebra Bio-PEPA with the possibility of assigning delays to actions following the delay-as-duration approach, yielding a new non-Markovian stochastic process algebra: Bio-PEPAd. Bio-PEPAd processes are defined with the original syntax of Bio-PEPA. We define process configurations, systems and we define a labeled semantics for Bio-PEPAd. We formally define the encoding of Bio-PEPAd models in Generalized semi-Markov Processes in input for the DDa and in sets of differential equations with delays. Also, we investigate the relation between Bio-PEPA and Bio-PEPAd models.

Finally, we give some conclusions and discuss further work in Chapter 10.

1.5 Published Material

Part of the material presented in this thesis has appeared in some publications or has been submitted for publication, in particular:

- The algorithm with delay-as-duration approach of SECTION 4.2, the one with purely delayed approach of SECTION 5.2, the stochastic and deterministic simulations of the cell cycle model of SECTIONS 3.2, 4.4, and 5.4 appear in (Barbuti *et al.*, 2009b).
- The algorithm with delayed propensity functions presented in SECTION 6.2, and its comparison with the PDA of SECTION 6.3 appears in (Barbuti *et al.*, 2011b).
- The definition of the purely delayed approach with markings of SECTION 7.2 and its combination with the delay-as-duration approach of SECTION 7.3 appear in (Barbuti *et al.*, 2011a).
- The definition of BIO-PEPAD of SECTIONS 9.2.1, 9.2.2 and 9.4.1 initially appeared in (Caravagna & Hillston, 2010). Moreover, an extended version including SECTIONS 9.2.3 and 9.3 appears in (Caravagna & Hillston, 2011).

All the published material is presented in this thesis in revised and extended form.

Extra material. Most of the models and the simulation software used to produce results outlined in this thesis, as well as the publications cited in the previous section, can be found at the web page of the *"Research Group on Modelling, Simulation and Verification of Biological Systems"* of the *Department of Computer Science* at the University of Pisa

`http://www.di.unipi.it/msvbio/`

or requested directly from the author at `caravagn@di.unipi.it`.

Chapter 2

Background

Since Systems Biology is an interdisciplinary field of research involving mathematics, biology and computer science at different level of detail, giving a comprehensive background of the required notions is a non trivial task. In this chapter we try to recall most of the background necessary to understand the results outlined in this thesis. Of course, this thesis is neither pure mathematics nor biology, so those parts are presented at a more introductory level. Moreover, the biological knowledge we need is recalled only when presenting models of target systems in CHAPTER 3. In the next, we assume the reader to be familiar with basic notions of calculus, which is something we need to discuss the mathematical foundations of the algorithms we present.

We start introducing notions of probability theory, random variables and some well-known probability distributions. We briefly introduce stochastic processes in their most common form, namely Markov processes, and a class of non-Markov processes useful to understand systems with non-instantaneous actions.

As far as deterministic models are concerned, we introduce differential equations both in delayed and non-delayed form. We introduce with a bit more detail stochastic models and an algorithm for simulating their time-evolution. We provide arguments about the mathematical foundations of such an algorithm and one of its generalization, as they are of help to discuss more complex results we present in the first part of the thesis.

As far as formal models are concerned, we recall basic notions to give a mathematical meaning to our languages and to the models we discuss. We introduce also techniques to define such mathematical objects for the languages we define in the second part of the thesis.

For all the concepts we recall, we discuss via simple examples their usability for our purposes.

2.1 Notions of Probability theory

In this section we recall some concepts from probability theory used along the thesis. Intuitively, probability is a way of measuring knowledge that an event will occur or has occurred. In probability theory, the *"probability of E"*, hereby denoted $\mathbb{P}(E)$, is defined so that $\mathbb{P}$ satisfies the *Kolmogorov* axioms. More precisely, a triple $(\Omega, F, \mathbb{P})$ is a *measure space* once that for any $E \in F$ it holds that $\mathbb{P}(E) > 0$, moreover $\mathbb{P}(\Omega) = 1$ and, for any sequence of pairwise disjoint events E_1, E_2, ..., E_n, it holds $\mathbb{P}(E_1 \cup \ldots \cup E_n) = \sum_{i=1}^{n} \mathbb{P}(E_i)$. So $(\Omega, F, \mathbb{P})$ is a *probability space*, with *sample space* Ω, *event space* F and *probability measure* $\mathbb{P}$. For the sake of our purposes, we introduce *discrete random variables* with $\mathbb{N}$ as sample space and *continuous random variables* with $\mathbb{R}$ as sample space.

If X is a discrete random variable there exist a *probability mass function* f characterizing the distribution of X and such that for any $x \in \mathbb{N}$ it holds $f(x) \geq 0$ and $\sum_{\mathbb{N}} f(i) = 1$. The probability of X having value x is $\mathbb{P}(X \leq x) = f(x)$ and, for any discrete random variable X we can define a

cumulative distribution function $F : \mathbb{N} \to [0, 1]$ such that

$$\mathbb{P}(X \leq x) = F(x) = \sum_{y \leq x} f(x)\,. \tag{2.1}$$

Similarly, if X is a continuous random variable there exist a *probability density function* f characterizing the distribution of X and such that for any $x \in \mathbb{R}$ it holds $f(x) \geq 0$ and $\int_{\mathbb{R}} f(x)dx = 1$. The probability of X having values in the closed real-valued interval $[a, b]$ is

$$\mathbb{P}(a \leq X \leq b) = \int_a^b f(x)dx\,.$$

In general, the probability of X being smaller than x is defined by the cumulative distribution function $F : \mathbb{R} \to [0, 1]$

$$\mathbb{P}(X \leq x) = F(x) = \int_{-\infty}^{x} f(u)du\,. \tag{2.2}$$

The cumulative distribution function F satisfies two intuitive equations

$$\lim_{x \to -\infty} F(x) = 0 \qquad\qquad \lim_{x \to +\infty} F(x) = 1\,.$$

Moreover, F is a non-decreasing function and if X is continuous in x then $F(x)$ is continuous, so we have that

$$\forall b \in \mathbb{R}.\, \mathbb{P}(X = b) = F(b) - \lim_{x \to b^-} F(x) = 0 \tag{2.3}$$

since the left-limit of F evaluates as $\lim_{x \to b^-} F(x) = F(b)$. Finally, it is easy to notice that

$$\mathbb{P}(X > x) = \int_{\mathbb{R}} f(u)du - \int_{-\infty}^{x} f(u)du = 1 - \mathbb{P}(X \leq x) = 1 - F(x)\,. \tag{2.4}$$

Before introducing some special cases of random variables, we introduce the notion of *conditioned probability*: $\mathbb{P}(A \mid B)$ represents the conditioned probability of "*A given B*" and is defined as

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A; B)}{\mathbb{P}(B)} \tag{2.5}$$

where $A; B$ denotes the event *"A and B"* and $\mathbb{P}(B) \neq 0$. Conditioned probability gives a way of measuring changes in the probability of an event A once we have some information about a related event B. When A and B are *independent* it holds that $\mathbb{P}(A; B) = \mathbb{P}(A)\mathbb{P}(B)$ and hence $\mathbb{P}(A \mid B) = \mathbb{P}(A)$.

A simple equation on conditioned probability can be immediately stated: given events A, B and C we have

$$\mathbb{P}(A \mid B; C)\mathbb{P}(B \mid C) = \mathbb{P}(A; B \mid C) \tag{2.6}$$

which can be easily verified by the definition of conditioned probability since

$$\mathbb{P}(A \mid B; C)\mathbb{P}(B \mid C) = \frac{\mathbb{P}(A; B; C)}{\mathbb{P}(B; C)} \frac{\mathbb{P}(B; C)}{\mathbb{P}(C)} = \frac{\mathbb{P}(A; B; C)}{\mathbb{P}(C)} = \mathbb{P}(A; B \mid C)\,.$$

The uniform distribution. Intuitively, a *continuous uniform distribution* is a family of distributions defined by lower and upper bounds a and b, respectively, such that all intervals of the same length in $[a, b]$ are equally probable. If X is uniformly distributed in $[a, b]$ we write

$$X \sim U[a, b]\,.$$

If $a = 0$ and $b = 1$ the resulting distribution $U[0,1]$ is called a *standard uniform distribution*. Analytically, the density characterizing $X \sim U[a,b]$ is defined as

$$f(x) = \begin{cases} (b-a)^{-1}, & a \leq x \leq b \\ 0, & x < a \wedge x > b \end{cases}$$

while the cumulative distribution function is

$$F(x) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ 1, & x > b\,. \end{cases}$$

A useful property of the standard uniform distribution is that

$$X \sim U[0,1] \implies (1-X) \sim U[0,1]\,.$$

The exponential distribution. A widely used continuous distribution in the context of systems biology is the *exponential distribution*. If X has exponential distribution with parameter $\lambda > 0$ we write

$$X \sim \mathcal{E}xp\,(\lambda)$$

and X has a probability density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0\,. \end{cases}$$

The cumulative distribution function is given by

$$F(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{-\lambda x}, & x \geq 0 \end{cases}$$

since

$$\int_0^x u e^{-\lambda u} du = 1 - e^{-\lambda x}\,.$$

The exponential distribution has *infinite support*, where the support of a distribution is the smallest closed interval whose complement has probability zero. Practically, this means that given any interval $[a,b]$ the probability of generating an exponentially distributed value in such interval is non zero, namely if $X \sim \mathcal{E}xp\,(\lambda)$ then

$$\forall a, b \geq 0.\ a < b \implies \mathbb{P}(a \leq X \leq b) \neq 0\,. \tag{2.7}$$

A very important property characterize solely this distribution: the *memoryless* property. If $X \sim \mathcal{E}xp\,(\lambda)$ then for any positive s, $t \in \mathbb{R}$

$$\mathbb{P}(X > s + t \mid X > t) = \mathbb{P}(X > s) \tag{2.8}$$

which holds since

$$\begin{aligned} \mathbb{P}(X > s+t \mid X > t) &= P(X > s+t; X > s) P(X > t)^{-1} \\ &= P(X > s+t) P(X > t)^{-1} \\ &= e^{-\lambda(s+t)} e^{\lambda t} = e^{-\lambda s}\,. \end{aligned}$$

Finally, let $\{X_i \sim \mathcal{E}xp\,(\lambda_i) \mid i = 1, \ldots, n\}$ where all the variables are independent, then let us define the new variable $Y = \min\{\, X_1, \ldots, X_n \,\}$. For such a variable we have

$$\mathbb{P}(Y > x) = \mathbb{P}(X_1 > x; \ldots; X_n > x) = \prod_{i=1}^{n} \mathbb{P}(X_i > x) = e^{-x \sum_{i=1}^{n} \lambda_i}$$

which means that

$$Y \sim \mathcal{E}xp\,(\lambda_1 + \ldots + \lambda_n)\,.$$

Sampling from the exponential distribution. We discuss now how to *generate* exponentially distributed numbers, an activity on which stochastic simulation is heavily based. The *sampling* of a value for a continuous random variable can be obtained by an *Inverse Monte-Carlo Algorithm* based on the following considerations: given a continuous random variable X with cumulative distribution F and given $p \sim U[0,1]$ it holds that

$$x = F^{-1}(p)\,. \tag{2.9}$$

Despite of being generally hard the computation of F^{-1} , for the exponential distribution we know how to evaluate the inverse of F. Let us assume $X \sim \mathcal{E}xp\,(\lambda)$, we start by noting that by definition of the exponential distribution

$$\mathbb{P}(X \leq x) = \int_0^x \lambda e^{-\lambda u} du$$

is the probability of X being smaller then x. Such value is a probability, so is a number in $[0,1]$; let us assume that we can pick a number $r \sim U[0,1]$. We can write

$$\int_0^x \lambda e^{-\lambda u} du = r$$

which integrates as

$$e^{-\lambda x} = 1 - r$$

and, as we know from the property of the uniform distribution, if $r \sim U[0,1]$ then $(1-r) \sim U[0,1]$. Computing now the value for x is fairly easy since when applying the logarithm we have

$$x = \lambda^{-1} \ln r^{-1}\,. \tag{2.10}$$

This last equation permits us to generate a sample for X once that we can pick a value for r.

The Erlang distribution. The exponential distribution is a special case of a more general continuous distribution, the *Erlang distribution*. A random variable X following such a distribution is denoted as

$$X \sim \Gamma(n, \lambda)$$

where $n \in \mathbb{N}$, $n > 0$ is called the shape, and $\lambda > 0$ is the rate. When $n \in \mathbb{R}$ this distribution is called the *Gamma distribution*. Erlang distribution has probability density function defined as

$$f(x) = \begin{cases} \dfrac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!}, & x \geq 0 \\ 0, & x < 0\,. \end{cases}$$

and cumulative distribution function defined as

$$F(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{-\lambda x} \sum_{i=0}^{n-1} \dfrac{(\lambda x)^i}{i!}, & x \geq 0\,. \end{cases}$$

Three important properties can be stated for this distribution: firstly, when the shape is 1 it reduces to the exponential distribution

$$X \sim \Gamma(1, \lambda) \implies X \sim \mathcal{E}xp\,(\lambda)$$

as it can be easily verified by the analytical form of the density function. Secondly, the summation of independent exponentially distributed random variables follows an Erlang distribution, namely

$$X_1 \sim \mathcal{E}xp\,(\lambda) \wedge X_2 \sim \mathcal{E}xp\,(\lambda) \implies (X_1 + X_2) \sim \Gamma(2, \lambda)\,.$$

and we also have that

$$X_1 \sim \Gamma(n_1, \lambda) \wedge X_2 \sim \Gamma(n_2, \lambda) \implies (X_1 + X_2) \sim \Gamma(n_1 + n_2, \lambda)$$

again if X_1 and X_2 are independent. Finally, it is easy to notice that this distribution has infinite support in the same sense as the exponential one.

2.2 Notions of Stochastic Processes

Here we recall some of the definitions of stochastic processes we use in the thesis, for a broad introduction to such a topic the reader can refer to (Ross, 1995). A stochastic process $\mathbf{X} = \{X(t), t \in T\}$ is a collection of random variables. Assuming T to represent *time*, if the set T is countable then the process is *discrete-time*, otherwise is *continuous-time*. Moreover, if $X(t)$ assumes discrete values, then the process is *discrete-state*, otherwise is *continuous-state*. In the context of this thesis we consider continuous-time and discrete-state stochastic processes with $T = \mathbb{R}$ and $X(t)$ assuming values on some discrete vector-space. In this sense, the discrete-state of the processes we consider represent *exact* numbers which, in our context of application, denote individuals in a system.

An important class of stochastic processes is the class of those satisfying the *Markov property*, namely the fact that given the present state of the process, its future is independent on the past. Not all the stochastic processes satisfy this property; in the next we introduce *Continuous-Time Markov Chains* (CTMCs) which satisfy such a property and *Generalized Semi-Markov Processes* (GSMP) which do not satisfy the Markov property.

Continuous-Time Markov Chains. Let us consider a continuous-time stochastic process $\mathbf{X} = \{X(t), t \in \mathbb{R}\}$ where the set of all possible values taken by $X(t)$ is a discrete set (e.g. the vector-space $\mathbb{N}^n$), we say that $\mathbf{X}$ is a CTMC if for any integer $k \geq 0$, sequence of time instants $t_0 < t_1 < \cdots < t_k$ and states $x_0, \ldots, x_k$ it satisfies the Markov property

$$\mathbb{P}(X(t_k) = x_k \mid X(t_{k-1}) = x_{k-1}, \ldots, X(t_1) = x_1) = \mathbb{P}(X(t_k) = x_k \mid X(t_{k-1}) = x_{k-1})\,. \tag{2.11}$$

Intuitively, this means that given the system in state x_{k-1} at time t_{k-1} the probability of moving to state x_k at time t_k depends only on the current state, and not on all the path starting in x_1 at time t_1 and ending up in the current state. Notice the similarity between the Markov Property and the memoryless property of the exponential distribution. Indeed, the exponential distribution is the only continuous probability distribution which exhibits such a property, hence it is the only one used in the definition of CTMCs.

Formally, a CTMC is defined as follows.

Definition (Continuous Time Markov Chain). A CTMC is a triple $\langle S, R, \pi \rangle$, where

- S is the finite set of states;
- $R : S \times S \to \mathbb{R}^{\geq 0}$ is the transition function;
- $\pi_0 : S \times S \to [0, 1]$ is the starting distribution.

The system is assumed to pass from a configuration modeled by a state $x \in S$ to another one modeled by a state $x' \in S$ by consuming an exponentially distributed quantity of time

$$\mathcal{E}xp\left(R(x, x')\right)\,.$$

This means that, by using the properties about the minimum of exponentially distributed random variables, the *sojourn time* in state x is distributed according to

$$\mathcal{E}xp\left(\sum_{x'' \in S} R(x, x'')\right)$$

where the summation $\sum_{x' \in S} R(x, x')$ is called the *exit rate* of state x. Practically this means that, when *jumping* between the states of the chain, all the outgoing transitions compete for being chosen forming a *race condition*, and the sojourn time in a state is driven by the quickest of its outgoing transitions. Whenever the sojourn time is established, among all the possible states reachable from x, the state x' the system moves to is chosen according to the weighted probability

$$\frac{R(x, x')}{\sum_{x'' \in S} R(x, x'')}\,.$$

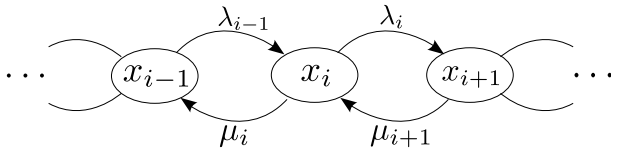

FIGURE 2.1: The CTMC for the linear birth-death process.

This last probability distributions is ruled by the *Discrete-Time Markov Chain* embedded in any CTMC, this embedded chain contains probabilistic information about the resolution of the transitions to fire in a state and, being discrete, it does not contain information about the distribution of the sojourn times.

Finally, the system is assumed to start from a configuration modeled by a state $x \in S$ with probability $\pi_0(x)$, and $\sum_{x \in S} \pi_0(x) = 1$. If the set of states of the CTMC is finite, $S = \{x_1, \ldots, x_n\}$, then the transition function R can be represented as a square *infinitesimal generator matrix* $\pi \in \mathbb{R}^{n \times n}$ such that

$$\pi_{i,j} = R(x_i, x_j) \qquad \pi_{i,i} = -\sum_{j \neq i} R(x_i, x_j)\,.$$

Some Markov chains are said to be *time-homogeneous* when the probability of a transition satisfies

$$\mathbb{P}(X(t_{k+1}) = x \mid X(t_k) = y) = \mathbb{P}(X(t_k) = x \mid X(t_{k-1}) = y)\,. \tag{2.12}$$

Such chains satisfy a desirable property once they have *finite* state space: the transition matrix is the same after each step, so the k-th step transition probability is the k-th power of the transition matrix, $\pi^k = \prod_{i=1}^{k} \pi$ where we assume the classic matrix multiplication. In this sense, a *stationary distribution* $\overline{\pi}$ is a row vector satisfying

$$\overline{\pi} = \overline{\pi}\pi\,. \tag{2.13}$$

Intuitively, the stationary distribution $\overline{\pi}$ is the fixed point of a linear transformation describing the *equilibrium* probability of the system, and under some conditions on π it is uniquely determined.

In order to clarify the use of CTMCs in modeling, let us consider a single population assuming discrete values in $\{n \in \mathbb{N} \mid n > 0\}$. In this population *death* and *birth* may happen, if the former happens population increases by one, conversely if the latter happens population decreases by one. We assume an infinite set of discrete states $S = \{x_i \mid x_i = i \wedge i \geq 0\}$; the transition function is such that

$$R(x_i, x_{i+1}) = \lambda_i \qquad R(x_i, x_{i-1}) = \mu_i$$

and $\forall i, j.\, R(x_i, x_j) = 0$ if $|i - j| > 1$. This means that, for instance, when the system is in state x_i it waits a quantity of time distributed according to $\mathcal{E}xp\,(\lambda_i + \mu_i)$

A graphical representation of this CTMC for the *linear birth-death process* is given in FIGURE 2.1. The values $\{\lambda_i \mid i \geq 0\}$ and $\{\mu_i \mid i \geq 1\}$ are called the birth and the death rates so when there are i people in the system the time until the next birth is exponential with rate λ_i and is independent of the time until the next death, which is exponential with rate μ_i.

Generalized Semi-Markov Processes. Using Markov processes is generally convenient since they are described by exponential functions which have a clear structure and are easy to understand. However, retrieving the Markov property for some systems may be hard or even unfeasible, in fact not all stochastic processes are Markov. Considering non-Markov processes is very hard since, in general, they are based on arbitrary distributions. However under some circumstances some considerations can be stated.

We introduce Generalized Semi-Markov Process (GSMP) as in (Bravetti *et al.*, 1998; Bravetti & Gorrieri, 2002), an extension of those originally defined in (Matthes, 1962). Such processes are

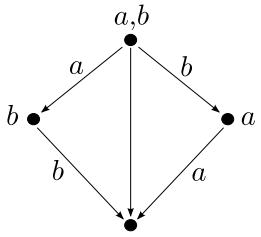

FIGURE 2.2: An example GSMP.

discrete processes, where the embedded state process is a Markov chain, but the time between jumps is a random variable of *arbitrary distribution*, which may be dependent on the two states between which the move is made. If in each state there is a single jump event, then the process is a *Semi-Markov Process*, in contrast to a GSMP which may have more than one event concurrently running in each state. If in a Semi-Markov process the times between jumps are exponentially distributed, namely jumps are memoryless, then it is a CTMC. GSMPs have been used to give a stochastic process description of a large class of discrete-event simulations (Cox, 1955).

Informally a GSMP is a process in which each state is characterized by a set of *active elements*, each with an associated *lifetime*. A state change occurs when an active element completes a lifetime and all interrupted elements record their *residual* lifetimes. Whenever the element is again active it *resumes* its remaining lifetime. If the lifetimes are exponential we may disregard the residual lifetimes, restarting each element with a new lifetime whenever it is active.

Definition (Generalized Semi-Markov Process) A *Generalized Semi-Markov Process* (GSMP) is defined on a set of states $\{x \mid x \in X\}$. For each x there are *active elements* s, from the set S, which decay at the rate $r(s, x)$, $s \in S$. When the active element s dies, the process moves to state $x' \in X$ with probability $p(x, s, x')$.

Another consideration is worth discussing. Markov processes, semi-Markov processes and GSMPs differ for the set of instants of process life which satisfy the Markov property, namely those instants such that the future behavior of the stochastic process depends only on the current state of the process and not on its past behavior. For Markov Chains the Markov property holds in every instant of process life, for Semi-Markov Processes it holds only in the instants of state change and, for a GSMP it never holds but can be retrieved through a different representation of process states. Such representation is given by turning each state into a continuous infinity of states by the standard technique discussed in (Cox, 1955).

We present now an example of GSMPs taken from (Bravetti & Gorrieri, 2002). Let us consider two activities a and b executed in parallel, both distributed with two arbitrary distributions. In FIGURE 2.2 the possible evolution of the two activities is given. In there, each state is labeled with the set of activities which are in execution during the period of time the system stays in the state. As expected, in the beginning both activities are together in execution and the system stays in the initial state until one activity or both contemporaneously terminates. When this happens the system performs the transition labeled with the terminated actions. If a terminates before b the system reaches the state labeled with b. In such a state the activity b continues its execution until it terminates. As a consequence the sojourn time of the system in the state labeled with b is given by the *residual* distribution of activity b, and is not determined simply by the fact that the system is in this state but depends on the time b has already spent in execution in the initial state. Clearly, this process is Markovian not even in the instant when this state is entered.

2.3 Deterministic Models of Biological Systems

Deterministic models of biological systems are the most used and widespread models of biological systems since the last century. In this section we introduce both the non-delayed and the delayed

deterministic frameworks for the modeling of biological systems. The former is characterized by *Ordinary Differential Equations* and the latter by *Delay Differential Equations*, a generalization of the ordinary ones.

2.3.1 Ordinary Differential Equations

A *Differential Equation* is an equation involving an unknown function and its derivatives; when the unknown function is a function of a single independent variable the equation is an Ordinary Differential Equation (ODE). There exist a variety of such equations, we concentrate on those useful in the context of our work. Typically, the models we consider are described as a set of ODEs which describes the time-evolution of the concentrations of the involved species in a given volume. Assuming the state of the system to be represented as a n-dimensional real-valued vector, the general form of an ODE for $\mathbf{X}(t) \in \mathbb{R}^n$ is

$$\frac{d\mathbf{X}}{dt} = f(t, \mathbf{X}(t)) \tag{2.14}$$

where $d\mathbf{X}/dt$ may depend on the state of the system at time t, denoted as $\mathbf{X}(t)$, and not on any previous states. Notice that, differently from CTMCs, in the context of systems biology the real-valued representation of $\mathbf{X}(t)$ is such that we do not represent exact numbers, but instead we represent *concentrations*.

Much study has been devoted to the solution of such equations; when the equation is *linear*, it can be solved by analytical methods. So for instance an equation of the form

$$\frac{dX}{dt} = \lambda X \tag{2.15}$$

has a well-known solution. Such an equation reads as "the variation in the size of X in the infinitesimal time dt is given by λX" and its analytical solution can be easily computed integrating the equation

$$\frac{dX}{X} = \lambda dt$$

and imposing the *initial condition* $X(t_0) = \mathbf{x}_0$. This means that the solution of the equation is

$$X(t) = X(t_0) \exp(\lambda t) \tag{2.16}$$

where $X(t_0)$ represent the initial value for X at time t_0. These kind of equations, which have always an exponential solutions, are said to model the *exponential decay/growth of* $X(t)$. In fact, once that the value for λ is known, if it is positive the solution models a growth, otherwise a decay.

Unfortunately, most of the interesting differential equations used in modeling biological systems are non-linear and, with a few exceptions, cannot be solved exactly. Approximate solutions are obtained by well-known numerical simulation algorithms: *Euler* forward and backward methods and *Runge-Kutta* iterative methods, to name but a few. Of course, this kind of analysis gives in general less information than the one given by analytical solutions. For an introduction to the theory of ODEs the reader is referred to (Agarwal & O'Regan, 2008).

In the context of biological systems ODEs have been used to study notions of chemical kinetics by means of *Reaction Rate Equations*, which are briefly introduced in the forthcoming section.

Chemical kinetics and Reaction Rate Equations

In this section we introduce some notions of *chemical kinetics* and we discuss how to define for some forms of chemical reactions their *Reaction Rate Equations* (RREs). For a broad introduction to the theory of chemical kinetics the reader is referred to (Segel, 1993).

Chemical kinetics is the theory which relates changes in the experimental conditions with the *rate* (or *frequency*) at which chemical reactions *fire*, also it relates such changes with the process of creating the results of the reaction. In order to fire a chemical reaction it is necessary that the

colliding reactants have a minimum level of activation energy and, if it is so, the firing of a reaction implies the break of the chemical bonds and the creation of new chemical bonds. These new chemical bonds, after some rearrangement processes leading to the creating of a stable chemical state, are such that the products of the reaction are present in the solution where the reaction fired.

Some of the main factors affecting the rate of a chemical reaction are the type and strength of the chemical bonds characterizing the reactants, the physical state of the medium, the quantity of the reactants and the average thermal energy of the system. As far as the quantity of the reactants is involved, the *collision theory* of chemical reactions states that the *probability of a collision between reactants in a medium is proportional to their concentrations.* Other quantities such as thermal energy of the molecules affect the rate as depicted by the *Maxwell-Boltzmann* equation.

The RRe for a chemical reaction is an ODe which links the reaction rate with some of the factors we discussed. Typically, the factors considered are concentrations of the reactants and constant coefficients. In order to define rate equations we assume a generic reaction of the form

$$l_1 R_1 + \ldots l_n R_n \overset{k}{\longmapsto} l'_1 P_1 + \ldots + l'_{n'} P_{n'}$$

transforming, for each reactant R_i with $i = 1, \ldots, n$, a number l_i of molecules and producing, for each product P_j with $j = 1, \ldots, n'$, a number l'_j of molecules. Notice that reactants and products can be mathematically represented as *multisets*, namely sets with repetitions. In that case, we would say that element R_i belongs to the multiset of reactants with multiplicity l_i, and element P_j belongs to the multiset of products with multiplicity l'_j. When we want to denote an empty multiset we use the symbol $\emptyset$.

The reaction is equipped with a *kinetic constant* $k \in \mathbb{R}$, a parameter relating temperature and other characterizing quantities of the target solution. Typically, chemical reactions are distinct dependently on the number of reactants they have: when they have zero ($n = 0$), one ($n = 1$) or two ($n = 2$) reactants they are considered 0-order, 1-order or 2-order reactions, respectively. Reactions involving more than 2 reactants can often be represented as multi-step 2-order reactions.

When the chemical kinetics is ruled by the *law of mass action* the reaction rate is proportional to the concentrations of the individual reactants involved. This definition assumes that the reaction happens in a homogeneous medium. Formally, such law states that the consumption and the production of the molecule involved in the reaction depends on the quantity

$$k \prod_{i=1}^{n} [R_i]^{l_i}$$

where $[R_i]$ represents the *concentration* of molecules R_i in the solution. From this quantity, a set of ODEs describing the changes induced by such reaction can be defined as

$$\frac{d[R_i]}{dt} = -k \prod_{i=1}^{n} [R_i]^{l_i} \qquad \frac{d[P_i]}{dt} = k \prod_{i=1}^{n} [R_i]^{l_i} \tag{2.17}$$

Intuitively, we define an ODe for each species appearing in the reaction, namely for any reactant and product. All these ODEs share a common term which is given by the law ruling the kinetics of the reaction. More precisely, this term is negative for the species involved as reactants, and is positive for those appearing as products and is both positive and negative for the combination of the cases.

As an example, let us consider two simple biochemical reactions

$$2A \overset{k_1}{\longmapsto} B \qquad A + B \overset{k_2}{\longmapsto} C$$

transforming two molecules A into the single molecule B with kinetic constant k, and transforming one molecule A and one molecule B in a molecule C with kinetic constant k'. By the law of mass

action we know that the rates at which the reaction happen are given by $k_1[A]^2$ and $k_2[A][B]$ and hence the ODEs which can be associated at the reaction are the following

$$\begin{aligned}\frac{d[A]}{dt} &= -k_1[A]^2 - k_2A][B] \\ \frac{d[B]}{dt} &= k_1[A]^2 - k_2[A][B] \\ \frac{d[C]}{dt} &= k_2[A][B]\,.\end{aligned}$$

More complex chemical kinetics laws have been defined (i.e. *Michaelis-Menten* or *Hill* kinetics) however, for the sake of simplicity, we do not recall their definition here.

2.3.2 Delay Differential Equations

In mathematics, Delay Differential Equations (DDEs) are equations in which the derivative of the unknown function at a certain time is given in terms of the values of the function at previous times. Consequently, DDEs are a more general framework than ODEs which result in being a particular case of DDEs in absence of delay. As typically happens, generalizing a framework implies the need of more complex analysis techniques. Indeed, the analysis either analytical or numerical of DDEs is more complex than the analogous for ODEs. Moreover, theoretical results about existence and uniqueness of solutions valid for ODEs do not hold for DDEs. For a broad introduction to DDEs and comparison with ODEs the reader is referred to (Driver, 1977; Smith, 2011).

The general form of a DDE for $\mathbf{X}(t)$ is

$$\frac{d\mathbf{X}}{dt} = f(t, \mathbf{X}(t), X_t), \tag{2.18}$$

where $X_t = \{\mathbf{X}(t') : t' \leq t\}$ represents the *trajectory of the solution in the past*; practically delays in DDEs are such that the derivative at time t depends on some past states of the system. Notice that this formulation is general enough to capture various forms of delays; we list and discuss some of the possible forms, ordered by complexity.

Constant delays. The simplest form of DDEs consider delays as constant real values, leading to equations of the form

$$\frac{d\mathbf{X}}{dt} = f(t, \mathbf{X}(t), \mathbf{X}(t-\sigma_1), \ldots, \mathbf{X}(t-\sigma_n)) \tag{2.19}$$

with $\sigma_1 > \ldots > \sigma_n \geq 0$, $\sigma_i \in \mathbb{R}$ and $\mathbf{X}(t-\sigma_i)$ denoting the state of the system at the past time $t-\sigma_i$. This form of DDE allows models to describe events which have a fixed constant duration since the delay is a point-wise dependency on a past-state of the system.

Variable delays. These DDEs have the form

$$\frac{d\mathbf{X}}{dt} = f(t, \mathbf{X}(t), \mathbf{X}(t-\tau_1(t, \mathbf{X}(t), P), \ldots, \mathbf{X}(t-\tau_n(t, \mathbf{X}(t), P)) \tag{2.20}$$

where $\tau_i(t, \mathbf{X}(t), P) : \mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}^+$ is a function of time, states and some physical parameter P. Similarly to the previous case delays are fixed point-wise dependency on past-states of the system, however in this case their values are not constant.

Distributed delays. The main feature of this form of delay is the dependency of a state on the full history of the system rather than, as in other forms, a point-wise dependency on past states. The general form of these DDEs is

$$\frac{d\mathbf{X}}{dt} = f\left(t, \mathbf{X}(t), \int_{-\infty}^{t} \gamma(t', \mathbf{X}(t'))dt'\right) \tag{2.21}$$

where $\int_{-\infty}^{t} \gamma(t', \mathbf{X}(t'))dt'$ is the integral over the time of a function $\gamma : \mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}^+$ of both time and state, hence represents dependency on the whole trace of the system.

All these different forms of delays can be used to model different kind of biological systems at different abstraction levels. In this thesis we concentrate on constant delays since their characterization in formal languages is non-trivial and can be used as a starting point for addressing more complex forms of delay.

Notice that, for any of the forms we presented, DDEs require a more complex notion of initial condition than ODEs. In fact, given an initial time t_0, the derivatives depends on instants preceding t_0 and hence the initial condition of DDEs is generally defined by means of functions, rather than a unique point as in ODEs. As an example of DDE, let us consider the equations

$$\frac{dX}{dt} = \lambda X(t - \sigma) \tag{2.22}$$

defined for $t \in [t_0, \infty)$, and with initial condition

$$X(t) = \phi(t) \tag{2.23}$$

defined for $t \in [t_0 - \sigma, t_0)$. Such a DDE is the equivalent of the ODE we previously presented, with the addiction of a delay. In fact, as expected it has the same kind of solutions, namely exponential growth or decay. What is unexpected is that this DDE has another type of solution which is *oscillatory*. The type of solution this equation has depends on the relationship between the parameters λ, σ and the function ϕ. One of the most interesting consequences of the correspondence between DDEs and ODEs is that DDEs exhibit in general different solutions, typically oscillatory where ODEs are not, as it happens in this very simple example. For results on oscillations in DDEs the reader is referred to (Driver, 1962; Arino *et al.*, 1984; Arino & Gyori, 1989; Baker *et al.*, 1999).

2.4 Stochastic Models of Biological Systems

In this section we introduce stochastic models of biological systems. Firstly, we discuss the utility of such systems as an alternative to deterministic ones. Secondly, we present an exact algorithm for analyzing such systems logically equivalent to a special ODE. Before presenting the results, we introduce a bit further notation.

Why stochastic models?

In this section we discuss the utility of stochastic models in combination with deterministic ones. We start by considering the deterministic version of the linear birth-death process we discussed in CTMCs, simplified assuming a unique rate for birth, and a unique rate for death. Such an example appears also in (Wilkinson, 2006) where a population of bacteria in a bacterial colony is to be modeled. In the colony, each bacterium produces λ offsprings per time unit and μ is the proportion of bacteria which die per time unit. These are to be rephrased as: each bacterium gives raise to new individuals at rate λ, and each bacterium dies at rate μ. Assuming to denote the colony with X, it is fairly intuitive that we can write the following linear ODE for the system

$$\frac{dX}{dt} = \lambda X - \mu X = (\lambda - \mu)X$$

which corresponds to the well-known form of ODE we previously introduced. As we discussed, for such an ODE the analytical solution is

$$X(t) = X(t_0) \exp\left((\lambda - \mu)t\right)$$

and by such a form of the solution the *qualitative* behavior summarized in TABLE 2.1 can be determined. The solution clearly depends only on the quantity $(\lambda - \mu)$ which means that once fixed such a value there exist infinite combinations of acceptable values for λ and μ which correspond

Condition	$\lim_{t\to\infty} X(t)$	Behavior
$\lambda > \mu$	$+\infty$	exponential growth
$\lambda = \mu$	$X(t_0)$	constant population
$\lambda < \mu$	0	exponential decay

TABLE 2.1: Qualitative behavior for the bacterial colony deterministic model.

to a unique deterministic behavior. Among all these pairs of values there are some which are surprisingly different; as an example consider $\lambda = 1$ and $\mu = 0.5$ compared to $\lambda = 0.5$ and $\mu = 0$. With these numbers in the former case we model a pure birth-death system, while in the latter a simple birth system since $\mu = 0$.

In a deterministic framework, knowing $(\lambda - \mu)$ implies knowing the whole dynamics of the system, even if we do not know the precise values for λ and μ. This is a good feature if we focus on model conciseness, and if continuity is appropriate for this model. In this sense, in the deterministic model the underlying simplistic assumption is that bacterial population varies in numbers continuously and deterministically.

However, in some cases we may require that $(\lambda - \mu)$ is not the only information we rely on, but we instead require precise values for both λ and μ. These parameters are crucial for performing inferences and predictions about biological networks of reactions. So for instance knowing whether we refer to birth-death or a birth process may give in general different inferences. If we want to move to this scenario, we have to require that bacteria vary stochastically particularly when the system is formed by small elements, so a proper model should include stochastic effects of the events therein. Stochastic models have the simple property of being deeply dependent on the value of the model variables so that each time they are analyzed they give possibly different behaviors. This effect is given by the inherent *stochastic fluctuations* of the model. Stochastic fluctuations can yield stochastic dynamics which have no deterministic counterparts, as discussed for instance in (Kouyous *et al.*, 2006; Caravagna *et al.*, 2010).

However, it is to be remarked that if the model is characterized by large numbers the stochastic fluctuations can be neglected with respect to the population size, and hence in those cases the deterministic approach is more reasonable and generally easier to be analyzed. Nowadays, the proper modeling strategy seems to be summarized by this criterion: if the model involves small numbers (few orders of magnitude) then the model should be stochastic, differently if it involves large numbers then it should be deterministic. Considerations supporting this criterion are discussed along the thesis.

Algebraic representation of chemically reacting systems

It is fairly convenient to think about chemically reacting systems as multisets rewriting systems (Banatre *et al.*, 1996; Barbuti *et al.*, 2009a). Similarly, it is possible to give a vector-based algebraic representation of chemically reacting systems (Gillespie, 1976). Both the representations are equivalent and define formal frameworks to define applicability of a chemical reaction or semantics of the firing of a chemical reaction. We introduce in this section the vector-based representation of biological systems since it is particularly convenient to formally define stochastic simulation of biological systems.

We consider a system of molecules belonging to N chemical species $\{S_1, \ldots, S_N\}$ interacting through M chemical reactions $R_1, \ldots, R_M$. We define for the target system an N-dimensional integer value vector $\mathbf{X}(t) \in \mathbb{N}^N$ such that

$$\mathbf{X}(t) = (X_1(t), \ldots, X_N(t))^T$$

is called the *state vector*. In there, we denote the *number of molecules* of species S_i in the system at time t with $X_i(t)$. In general, we may write $\mathbf{X}(t) = \mathbf{x}$ to precisely characterize the time-dependent vector $\mathbf{X}(t)$.

We assume each species to be mapped to a unique location in the vector by means of a bijective mapping; this gives that a single location refers to a single species, and viceversa. Defining such a mapping would be easy and for the sake simplifying the notation we omit its definition; for instance, in the examples we present we generally assume such a mapping to be the lexicographic ordering of the species names.

A reaction R_j is algebraically represented by its associated *state-change vector* $\nu_j \in \mathbb{N}^N$

$$\nu_j = (\nu_{1,j}, \ldots, \nu_{N,j})^T$$

where $\nu_{i,j}$ is defined to be the change in the S_i molecular population caused by one R_j reaction. If we consider the structure of a chemical reaction we informally say that some molecules are *created* and others *destroyed* by the reaction. More precisely, if in a reaction w molecules of a species appear as reactants and w' appear as products, then if $w > w'$ we say that $|w' - w|$ molecules are consumed, if $w < w'$ we say that $(w' - w)$ molecules are created and, finally, if $w = w'$ the reaction does not affect the species. Accordingly to this consideration the state-change vector is defined as $\nu_{i,j} = (w' - w) = \Delta w$ so that we have

$$\nu_{i,j} = \begin{cases} -\Delta w, & \text{if } R_j \text{ consumes } \Delta w \text{ molecules of species } X_i(t) \\ 0, & \text{if } R_j \text{ does not affect species } X_i(t) \\ \Delta w, & \text{if } R_j \text{ creates } \Delta w \text{ molecules of species } X_i(t)\,. \end{cases}$$

From the state-change vector of the reactions it is possible to define the *stoichiometry matrix* $D \in \mathbb{N}^{N \times M}$

$$D = \begin{bmatrix} \nu_1 & \nu_2 & \ldots & \nu_M \end{bmatrix}\,.$$

A consequence of this algebraic representation is that the semantics of firing a chemical reaction turns out to be fully represented as simple vector summation. Indeed, given $\mathbf{X}(t) = \mathbf{x}$ and a reaction R_j the firing of the reaction modifies the state accordingly to this simple equation

$$\mathbf{x}' = \mathbf{x} + \nu_j \tag{2.24}$$

where $\mathbf{x}'$ is the new state vector where the reactants are removed and the products inserted, as can be easily verified. We clarify this notation with a simple example: we consider molecules of species A, B and C, and two reactions

$$R_1 : A + B \overset{k_1}{\longmapsto} C \qquad\qquad R_2 : C \overset{k_2}{\longmapsto} C + B\,.$$

For this system, the state vector $\mathbf{X}(t_0) = \mathbf{x}_0$ and the state-change vectors can be represented as

$$\mathbf{x}_0 = \begin{pmatrix} n_A \\ n_B \\ n_C \end{pmatrix} \qquad \nu_1 = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \qquad \nu_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

where n_A, n_B and n_c represent the number of molecules of species A, B and C. The state-change vector ν_1 models the changes induced by the firing of reaction R_1, indeed the first component of the vector, which refers to species A, is -1 since one molecule A is consumed. Differently, the third component of vector ν_2, which models the changes induced by the firing of reaction R_2, is 0 since one molecule C is consumed/produced at the same time, namely it appears as a reactant and a product. For such a system the stoichiometry matrix is defined as $D \in \mathbb{N}^{3 \times 2}$

$$D = \begin{bmatrix} \nu_1 & \nu_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -1 & -1 \\ 1 & 0 \end{bmatrix}\,.$$

The sequential firing of reaction R_1 and R_2 changes the state vector $\mathbf{X}(t_0)$ as

$$\mathbf{X}(t_1) = \mathbf{x}_0 + \nu_1 = \begin{pmatrix} n_A \\ n_B \\ n_C \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} n_A - 1 \\ n_B - 1 \\ n_C + 1 \end{pmatrix} = \mathbf{x}_1$$

Type	Reaction	Propensity
zero order	$\emptyset \overset{k}{\longmapsto} B$	k
first order	$A \overset{k}{\longmapsto} B$	$k[A]$
second order	$2A \overset{k}{\longmapsto} B$	$k[A]([A]-1)/2$
	$A_1 + A_1 \overset{k}{\longmapsto} \mathrm{B}$	$k[A_1][A_2]$

TABLE 2.2: Analytical form of the propensity functions.

and as

$$\mathbf{X}(t_2) = \mathbf{x}_1 + \nu_2 = \begin{pmatrix} n_A - 1 \\ n_B - 1 \\ n_C + 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} n_A - 1 \\ n_B \\ n_C + 1 \end{pmatrix}$$

where t_1 and t_2 are the times at which reactions R_1 and R_2 fire, respectively.

2.4.1 The Chemical Master Equation

In order to introduce algorithms for the simulation of biological systems we must firstly discuss their mathematical basis. Indeed, in this section we recall the definition of the *Chemical Master Equation* (CME) as originally in (Gillespie, 1976; Gillespie, 1977). Technically, the CME is a *Partial Differential Equation* representing a set of ODEs describing the time-evolution of the probability of a system to occupy each one of a set of states.

We consider a system described by a state vector $\mathbf{X}(t)$ and M reactions; we study the time-evolution of $\mathbf{X}(t)$ assuming that the system was initially in some state $\mathbf{X}(t_0) = \mathbf{x}_0$ at time t_0. From the physical point of view we assume the molecules in the system to be *well-stirred* so that their positions become randomly uniform over a contained volume. We also assume the system to be confined in a constant volume and to be in thermal equilibrium at some constant temperature.

The notion of propensity function. Accordingly to (Gillespie, 1976; Gillespie, 1977), besides the algebraic representation of each reaction we associate a *propensity function* $a_j(\mathbf{x})$ to each R_j so that $a_j(\mathbf{x})dt$, given $\mathbf{X}(t) = \mathbf{x}$, is the probability of reaction R_j to fire in state $\mathbf{x}$ in the next infinitesimal time $[t, t + dt)$. The general definition of propensity function depends on the reaction we are modeling; TABLE 2.2 summarizes the analytical form of the propensity functions for chemical reactions, as originally defined in (Gillespie, 1977). In such a table $[A]$ denotes the *number of molecules* A in the system state, hence $X_i(t)$ if species A is assigned to location i in $\mathbf{X}(t)$. It is fairly easy to notice that well-stirred assumption gives rise to the combinatorial form of such functions. Moreover, although generalizing the analytical form for the propensity functions for reactions of any order is possible, we remark that most higher order reactions can be decoupled in sequences of second order reactions. Finally, it is important to notice that for the reactions in TABLE 2.2 we have that if $[A] = 0$ then the appropriate propensity function evaluates to 0, which practically defines the non-applicability of the reaction in the current state because of the absence of the reactants. Of course, this holds only for reactions requiring non-empty reactants, so in general only for first order and second order ones.

With respect to the example we previously discussed, to reactions $R_1 : A + B \overset{k_1}{\longmapsto} C$ and $R_2 : C \overset{k_1}{\longmapsto} C + B$ it is possible to associate the following propensity functions

$$a_1(\mathbf{x}) = k_1 X_1(t) X_2(t) \qquad\qquad a_2(\mathbf{x}) = k_2 X_3(t)$$

for state vector $\mathbf{X}(t) = \mathbf{x}$ since, species A, B and C are mapped to locations 1, 2 and 3 in the state vector, respectively.

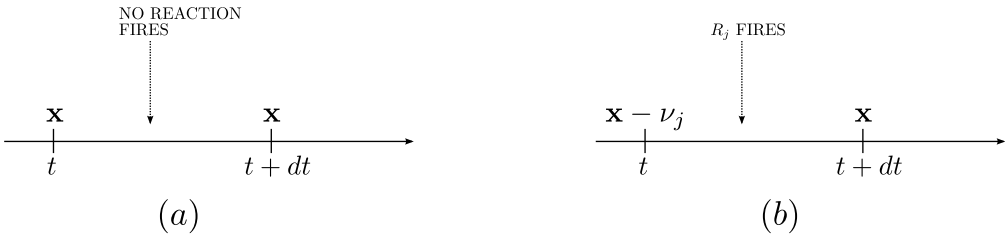

FIGURE 2.3: Events leading to the definition of the CME.

Construction of the CME. The CME is a set of ODEs describing the time evolution of the probability of a system to occupy each one of a discrete set of states. In its original definition we denote with

$$\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)$$

the probability that, given the initial configuration $\mathbf{X}(t_0) = \mathbf{x}_0$, at time t the system is described by the state vector $\mathbf{x}$, namely $\mathbf{X}(t) = \mathbf{x}$. The CME is the differential equation describing the variation of such a probability in the infinitesimal time dt. In this sense, the fact that the CME defines an unknown probability function by means of its derivative is the conceptual connection among deterministic and stochastic models.

In order to define $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)$ we use its analogous at time $t + dt$, namely $\mathbb{P}(\mathbf{x}, t + dt \mid \mathbf{x}_0, t_0)$. Such a probability, assuming that dt is chosen so small that at most one reaction fires in the time interval $[t, t + dt)$, is defined in terms of these two events:

(a) at time t the system is in state $\mathbf{x}$ and in the infinitesimal time $[t, t + dt)$ no reaction fires;

(b) at time t the system is in state $\mathbf{x} - \nu_j$ and reaction R_j fires.

In FIGURE 2.3 the events leading to the definition of the CME are graphically represented. We discuss now the analytical definition of the probability of these events.

No reactions fire, event (a)**.** The probability of having no state changes in $[t, t + dt)$ is given by the probability of the system to be in state $\mathbf{x}$ and the negation of the event that at least one of the reactions fires. These events are independent and the probability of the former is by definition $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)$. Moreover, the probability of firing any reaction, $\sum_{j=1}^{M} a_j(\mathbf{x})dt$, defines the probability of the latter as $1 - \sum_{j=1}^{M} a_j(\mathbf{x})dt$. The conjunction of the independent events leads to the definition of the probability of the target event as

$$\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0) \left(1 - \sum_{j=1}^{M} a_j(\mathbf{x})dt \right) .$$

Reaction R_j **fires, event** (b)**.** The probability of event (b) is to be defined accordingly to the generalization, among all the reactions, of the probability of being in state $\mathbf{x} - \nu_j$ at time t and to fire, in that particular state, reaction R_j. Such an event for a single reaction has probability $\mathbb{P}(\mathbf{x} - \nu_j, t \mid \mathbf{x}_0, t_0) a_j(\mathbf{x} - \nu_j)$, which is generalized as

$$\sum_{j=1}^{M} \mathbb{P}(\mathbf{x} - \nu_j, t \mid \mathbf{x}_0, t_0) a_j(\mathbf{x} - \nu_j)dt \, .$$

Such an event is correct since we know that, by applying R_j, the system moves from $\mathbf{x} - \nu_j$ to $\mathbf{x} - \nu_j + \nu_j$.

Derivation of the CME. The whole probability $\mathbb{P}(\mathbf{x}, t+dt \mid \mathbf{x}_0, t_0)$, obtained by summation of the probability of events (a) and (b), is given by

$$\mathbb{P}(\mathbf{x}, t+dt \mid \mathbf{x}_0, t_0) = \mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)\left(1 - \sum_{j=1}^{M} a_j(\mathbf{x})dt\right) + \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}-\nu_j, t \mid \mathbf{x}_0, t_0) a_j(\mathbf{x}-\nu_j)dt$$

which rewrites as

$$\frac{\mathbb{P}(\mathbf{x}, t+dt \mid \mathbf{x}_0, t_0) - \mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)}{dt} = -\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0) \sum_{j=1}^{M} a_j(\mathbf{x}) + \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}-\nu_j, t \mid \mathbf{x}_0, t_0) a_j(\mathbf{x}-\nu_j)\,.$$

When we consider the limit $dt \to 0$ we get the CME

$$\frac{\partial \mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)}{\partial t} = \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}-\nu_j, t \mid \mathbf{x}_0, t_0) a_j(\mathbf{x}-\nu_j) - \mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0) a_j(\mathbf{x}). \tag{2.25}$$

Equation (2.25) is a *Partial Differential Equation* since we derived only with respect to t, whereas $\mathbf{x}$ is a variable as well; once $\mathbf{x}$ is fixed this becomes an ODE. However, as argued in (Gillespie, 1977) the CME is generally difficult to solve, in particular it can be solved analytically only for a very few simple systems and numerical solutions may be prohibitively difficult. These difficulties motivated in defining alternative techniques for finding its solution, leading to the definition of stochastic models. More precisely, stochastic simulation algorithm can be defined from the CME so that the probability density function of the defined stochastic process is the exact solution of the CME.

2.4.2 The Stochastic Simulation Algorithm (SSA)

In this section we recall the definition of the *exact Stochastic Simulation Algorithm* (SSA) (Gillespie, 1976; Gillespie, 1977) in its *Direct Method* formulation. Although equivalent exact formulations of the SSA exist such as the *First Reaction Method* or the method proposed in (Gibson & Bruck, 2000), we refer to the Direct Method formulation as "the SSA". For approximations of exact methods we refer to (Gillespie, 2001; Rathinam *et al.*, 2003; Cao *et al.*, 2005). For a good introduction to stochastic modeling the reader is referred to (Gillespie & Petzold, 2006; Wilkinson, 2006)

The SSA is an exact *Dynamic Monte-Carlo Method* describing a statistically correct trajectory of a discrete non-linear Markov process, whose probability density function is the solution of the CME of equation (2.25). The SSA deals with the problem of computing a *single* realization of the process $\mathbf{X}(t)$, which is not not equivalent to solving the CME as sometimes misunderstood. The numerical solution of the CME, namely the probability distribution associated to $\mathbf{X}(t)$, could be obtained by sampling and averaging *infinite* realizations of SSA runs (Gillespie & Petzold, 2006).

The algorithm allows a discrete and stochastic simulation of a system with small number of reactants since every reaction is explicitly simulated. We start by describing the algorithm and, at the end of the section, we provide arguments on its strong mathematical foundation. Formally, the algorithm simulates the time-evolution of a system described by a state vector $\mathbf{X}(t) = \mathbf{x}$ and in which a set of reactions $\mathcal{R}$ can happen. Its formal definition is ALGORITHM 1.

The SSA assumes as input an initial simulation time t_0, a maximum simulation time T and an initial state $\mathbf{x}_0$ such that $\mathbf{X}(t) = \mathbf{x}_0$. At each step of the algorithm two decisions are taken: *when* is going to fire the next reaction and *which* reaction it will be. Given the system in state $\mathbf{x}$

Algorithm 1 SSA $(t_0, \mathbf{x}_0, T)$

1: $t \leftarrow t_0$;
2: $\mathbf{x} \leftarrow \mathbf{x}_0$;
3: **while** $t < T$ **do**
4: $\quad a_0(\mathbf{x}) \leftarrow \sum_{j=1}^{M} a_j(\mathbf{x})$;
5: $\quad$ let r_1, $r_2 \sim U[0,1]$;
6: $\quad \tau \leftarrow a_0(\mathbf{x})^{-1} \ln({r_1}^{-1})$;
7: $\quad$ let j such that $\sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$;
8: $\quad \mathbf{x} \leftarrow \mathbf{x} + \nu_j$;
9: $\quad t \leftarrow t + \tau$;
10: **end while**

at time t, the *putative time τ for the next reaction to fire* is chosen by sampling an exponentially distributed random variable such that

$$\tau \sim \mathcal{E}xp\,(a_0(\mathbf{x}))$$

where $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$. The sampling of such variable in step (6) is obtained by the Inverse Monte-Carlo Algorithm discussed in SECTION 2.1 by using uniformly distributed numbers r_1 and r_2, generated at step (5). Once τ is sampled, another random variable with values in $\{1, \ldots, M\}$ denoting *which* is the reaction to fire at time $t+\tau$ is sampled in step (7) accordingly to the following inequalities

$$\sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$$

which model a probabilistic choice dependent on the evaluations of the propensity functions for the M reactions. Again, this means that every reaction is chosen with weighted probability $a_j(\mathbf{x})/a_0(\mathbf{x})$, in fact another formulation for such choice is given by $j = \min\{n \mid r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{n} a_i(\mathbf{x})\}$. When both the variables have been sampled, the system state is updated performing the firing of R_j and setting time to $t + \tau$, as given by steps (8) and (9).

Notice that, even if it may seem an intuitive interpretation that the values for τ represent the durations of the reactions (i.e. a reaction starts firing at time t and completes at time $t + \tau$), this interpretation turns out to be confusing when introducing the notions of delay in stochastic simulation algorithms. In fact, it turns out from the discussion on the mathematical foundations of the SSA, that the values of τ represent time instants in which the system state is left unchanged. Indeed, the correct interpretation is that, with the system at time t, the system is left unchanged in $[t, t + \tau)$ and then performs an *instantaneous* change by firing a reaction at time $t + \tau$.

Mathematical foundations of the SSA. The SSA is a quite simple algorithm whose mathematical foundations can be precisely investigated. In particular, there are two points which have to be discussed: why the putative time for the next reaction to fire is an exponential random variable, and why the reaction to fire is chosen with weighted probability.

Given $\mathbf{X}(t) = \mathbf{x}$, let us denote the probability of the next reaction to fire at time $t + \tau$ as $p(\tau, j \mid \mathbf{x}, t)$, and the probability that reaction to fire is R_j, given it is going to fire at $t + \tau$, as $P(j \mid \tau; \mathbf{x}, t)$. The former is a probability density function of a continuous random variable assuming values in $[0, \infty)$, and the latter is a probability mass function of a discrete random variable assuming values in $[0, M]$. The algorithm is correct if and only if the continuous random variable turns out to be exponentially distributed, and the discrete random variable to have weighted probability dependent on the propensity functions.

Results presented in (Gillespie, 1976) provide the mathematical correctness of the SSA.

THEOREM 2.4.1 (SSA Correctness)**.** *The putative time for the next reaction is a continuous random variable $\tau \sim \mathcal{E}xp\,(a_0(\mathbf{x}))$ since*

$$p(\tau \mid \mathbf{x}, t) = a_0(\mathbf{x}) \exp\,(-a_0(\mathbf{x})\tau) \tag{2.26}$$

and the index of the next reaction to fire is a discrete random variable j with

$$P(j \mid \tau; \mathbf{x}, t) = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}\,. \tag{2.27}$$

Proof. As stated in (Gillespie, 1976; Gillespie & Petzold, 2006), the key in generating simulated trajectories for $\mathbf{X}(t)$ is not the CME or even the function $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0)$, but rather the new function $p(\tau, j \mid \mathbf{x}, t)$ defined such that $p(\tau, j \mid \mathbf{x}, t)d\tau$ is the probability, given $\mathbf{X}(t) = \mathbf{x}$, that the next reaction in the system occurs in the infinitesimal time interval $[t + \tau, t + \tau + d\tau)$ and is reaction R_j. Notice that our target density function $p(\tau \mid \mathbf{x}, t)$ can be defined from a generalization of $p(\tau, j \mid \mathbf{x}, t)$ among all the possible reactions. Formally, $p(\tau, j \mid \mathbf{x}, t)$ is the joint probability density function of two random variables, one modeling the putative time for the next reaction (τ), and the other modeling the choice of the next reaction to fire (j). It is important how an analytical expression for such function can be derived.

Firstly, we start by denoting with $P_0(\tau \mid \mathbf{x}, t)$ the probability that no reaction fire in the time interval $[t, t + \tau)$, so that we can write $p(\tau, j \mid \mathbf{x}, t)$ as

$$p(\tau, j \mid \mathbf{x}, t) = P_0(\tau \mid \mathbf{x}, t)a_j(\tau)d\tau \tag{2.28}$$

where we write $a_j(\tau)$ to denote that we have to evaluate the propensity function of the reaction in the state at time $t + \tau$. Notice that by this formula it is clear that the correct interpretation for the values of τ is that they represents instants in which the system does not change. At this point, we are expected to have an analytical expression for $P_0(\tau \mid \mathbf{x}, t)$; we can define, similarly as we did for the CME, an ODE for $P_0(\tau \mid \mathbf{x}, t)$. Formally, we can write

$$P_0(\tau + d\tau \mid \mathbf{x}, t) = P_0(\tau \mid \mathbf{x}, t)(1 - a_0(\tau)d\tau)$$

where $a_0(\tau)$ is the sum of all the propensity functions evaluated in the state of the system at time $t + \tau$. When considering $d\tau \to 0$, we get the ODE

$$\frac{dP_0(\tau \mid \mathbf{x}, t)}{d\tau} = -P_0(\tau \mid \mathbf{x}, t)a_0(\tau)\,. \tag{2.29}$$

This equation models the exponential decay of $P_0(\tau \mid \mathbf{x}, t)$ and its solution is

$$P_0(\tau \mid \mathbf{x}, t) = P_0(0 \mid \mathbf{x}, t) \exp\left(-\int_0^\tau a_0(\tau)d\tau\right)\,.$$

Two considerations can be done now: the first is that the initial condition $P_0(0 \mid \mathbf{x}, t) = 1$ since it represents the probability that nothing happens in 0 time. The second is that, by the definition of τ as a time interval in which nothing happens, and this is true in these systems since reactions are instantaneous, so $\forall t' \in [t, t + \tau).\, a_0(t') = a_0(\mathbf{x})$, and hence $a_0(\tau)$ does not depend on τ, then we have that

$$P_0(\tau \mid \mathbf{x}, t) = \exp\left(-a_0(\mathbf{x})\tau\right) \tag{2.30}$$

since $-\int_0^\tau a_0(\tau)d\tau = -a_0(\mathbf{x})\int_0^\tau d\tau$. Equation (2.28) can be rewritten by means of equation (2.30)

$$p(\tau, j \mid \mathbf{x}, t) = a_j(\mathbf{x})d\tau \exp\left(-a_0(\mathbf{x})\tau\right)$$

and then can be generalized among all the possible reactions in order to get the analytical definition of equation (2.26),

$$p(\tau \mid \mathbf{x}, t) = \sum_{j=1}^{M} p(\tau, j \mid \mathbf{x}, t) = a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right)\,.$$

This equation implies that the values for τ in the SSA must be exponentially distributed with mean $a_0(\mathbf{x})$ and, consequently, justifies step (6) of the algorithm. Finally, the correctness in generating

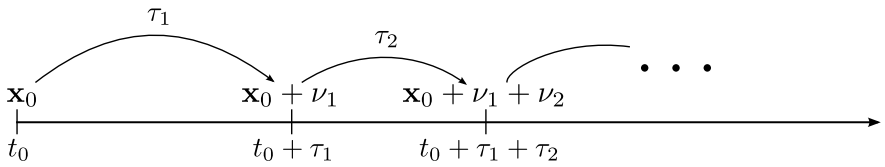

FIGURE 2.4: An example SSA computation.

the index of the next reaction to fire comes from the definition of the probability mass function for j; we can argue that the probability to fire R_j at $t+\tau$, given that one reaction is to be fired at such time, is defined as the conditional probability

$$P(j \mid \tau; \mathbf{x}, t) = \frac{p(\tau, j \mid \mathbf{x}, t)}{p(\tau \mid \mathbf{x}, t)} = \frac{a_j(\mathbf{x}) \exp(-a_0(\mathbf{x})\tau)}{a_0(\mathbf{x}) \exp(-a_0(\mathbf{x})\tau)} = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}.$$

which gives the analytical definition of equation (2.27). □

This theorem provides two important results: equations (2.26) and (2.27) provide the correctness of steps (6) and (7), respectively, and the SSA and the CME are *logically equivalent* since they are built by using the same assumptions. Furthermore, we remark that this algorithm is exact in the sense that produces one exact trajectory in the state space of the system, among all the possible trajectories starting from $\mathbf{X}(t_0) = \mathbf{x}_0$. Other algorithms for the stochastic simulation of biological systems as the method in (Gibson & Bruck, 2000) or the First Reaction Method (Gillespie, 1977) are based on the ideas outlined here and their correctness is proved by similar arguments.

An example computation. Let us consider a system described by an initial state $\mathbf{x}_0$ and two reactions R_1 and R_2. We assume the initial state $\mathbf{x}_0$ to be such that the reactions can fire an arbitrarily amount of times.

We show now some steps of the computation SSA(t_0, $\mathbf{x}_0$, T) where $T > t_0$. To shorten the notation, we use $\mathbf{X}(t') = \mathbf{x}'$ to denote the assignments of the variables $t \leftarrow t'$ and $\mathbf{x} \leftarrow \mathbf{x}$. Initially, the propensity functions are evaluated so that $a_0(\mathbf{x}_0) = a_1(\mathbf{x}_0) + a_2(\mathbf{x}_0)$ and the putative time for the next reaction to fire is generated as $\tau_1 \sim \mathcal{E}xp(a_0(\mathbf{x}_0))$; now each of the reaction is chosen to fire with probability either $a_1(\mathbf{x}_0)/a_0(\mathbf{x}_0)$ or $a_2(\mathbf{x}_0)/a_0(\mathbf{x}_0)$. If R_1 is chosen we have $\mathbf{X}(t_0+\tau_1) = \mathbf{x}_0+\nu_1$, otherwise $\mathbf{X}(t_0 + \tau_1) = \mathbf{x}_0 + \nu_2$.

We assume to fire reaction R_1 and $t_0+\tau_1 < T$. In the next step of the algorithm the propensity functions are evaluated so that $a_0(\mathbf{x}_0 + \nu_1) = a_1(\mathbf{x}_0 + \nu_1) + a_2(\mathbf{x}_0 + \nu_1)$ and the putative time for the next reaction to fire is generated as $\tau_2 \sim \mathcal{E}xp(a_0(\mathbf{x}_0 + \nu_1))$; again each reaction is chosen to fire with probability either $a_1(\mathbf{x}_0 + \nu_1)/a_0(\mathbf{x}_0 + \nu_1)$ or $a_2(\mathbf{x}_0 + \nu_1)/a_0(\mathbf{x}_0 + \nu_1)$. If R_1 is again chosen we have $\mathbf{X}(t_0 + \tau_1 + \tau_2) = \mathbf{x}_0 + 2\nu_1$, otherwise $\mathbf{X}(t_0 + \tau_1 + \nu_2) = \mathbf{x}_0 + \nu_1 + \nu_2$. A graphical representation of this sequences of steps for the SSA is given in FIGURE 2.4, in there R_1 fires first, and R_2 second.

2.4.3 SSA-based algorithms with time-dependent propensity functions

In this section we recall a result appearing in (Shahrezaei *et al.*, 2008) which is useful in the rest of the thesis. In there, the authors present a SSA-based algorithm with time-dependent propensity functions allowing continuous and discontinuous changes in reaction rates to simulate extrinsic and intrinsic fluctuations.

Assuming the system to be in state $\mathbf{x}$ at time t, they consider reactions with time-dependent propensity functions denoted as $a_j(\tau)$ (i.e. the propensity function of reaction R_j is evaluated at time $t + \tau$) and as $a_0(\tau) = \sum_{j=1}^{M} a_j(\tau)$. Notice that this is a more general framework than the one assumed in the SSA. Indeed, we know that in the SSA $a_j(\tau)$ evaluates as $a_j(\mathbf{x})$ because of the considerations we previously stated. Differently, here it is not specified whether the propensity functions depend on some states $\mathbf{x}'$ at time $t' \neq t$, but it is simply assumed that the functions

depend on time. Now let us assume that the time-dependent propensity functions are piece-wise constant, namely it is possible to determine a time instant φ such that $a_0(t')$ is constant for all the time window $[t, t+\varphi)$. Indeed, let us write this recursive definition for $a_0(t')$

$$a_0(t') = \begin{cases} \xi & \text{if} \quad t \leq t' < t+\varphi \\ a_0(t') & t' \geq t+\varphi\,. \end{cases}$$

From this very general assumption we can get a general schema of SSA-based algorithm which we use in the next chapters of the thesis. We start by considering systems such that the probability of the next event to occur at time τ is defined as

$$p(\tau \mid \mathbf{x}, t) = a_0(\tau) \exp\left(-\int_0^\tau a_0(t')dt'\right)\,. \tag{2.31}$$

Notice that if we rephrase this equation in the SSA we have equation (2.26) modeling the probability of the next reaction to fire at time $t+\tau$ to be expressed as

$$p(\tau \mid \mathbf{x}, t) = a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right)\,.$$

We can now prove the following theorem.

THEOREM 2.4.2. *The probability of the next event to occur at time τ satisfies*

$$p(\tau \mid \mathbf{x}, t) = \begin{cases} \xi \exp\left(-\xi\tau\right) & \textit{if} \quad 0 \leq \tau < \varphi \\ p(\tau-\varphi \mid \mathbf{x}', t+\varphi) & \tau \geq \varphi\,. \end{cases} \tag{2.32}$$

Proof. In typical SSA-based algorithms we need to sample values for τ from $p(\tau \mid \mathbf{x}, t)$. Let us assume a sample of a uniformly distributed number $r \sim U[0,1]$, the Inverse Monte-Carlo Algorithm discussed in SECTION 2.1 is based on solving the following equation

$$\int_0^\tau p(t' \mid \mathbf{x}, t)dt' = r$$

which rewrites as

$$\int_0^\tau dt' a_0(t') \exp\left(-\int_0^{t'} a_0(t'')dt''\right) = r$$

and can be solved as

$$\exp\left(-\int_0^\tau a_0(t')dt'\right) = 1-r$$

since if $r \sim U[0,1]$ then also $(1-r) \sim U[0,1]$. If we want to try to solve this equation for τ, we can firstly rewrite everything as

$$\int_0^\tau a_0(t')dt' = \log(r^{-1})\,. \tag{2.33}$$

Now, the assumption on the time-dependent propensity functions as piece-wise constant functions combined with equation (2.33) implies that r and the logarithmic term $\log(r^-1)$ are inversely proportional, so if r is big enough having a value r_1, if $\tau < \varphi$, we can rewrite the equation as

$$\int_0^\tau \xi dt' = \log({r_1}^{-1}) \tag{2.34}$$

which is the classical SSA scheme to generate a sample for τ as $\xi^{-1}\log({r_1}^{-1})$. Differently, if r is small having a value r_2 we need to solve, for $\tau > \varphi$, equation

$$\int_0^\varphi \xi dt' + \int_0^{\tau-\varphi} a_0(t')dt' = \log({r_2}^{-1})\,.$$

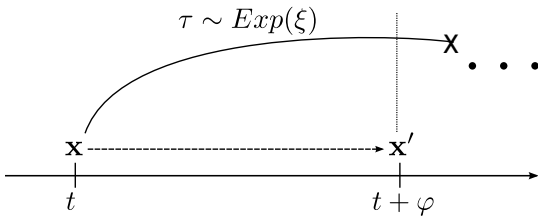

FIGURE 2.5: The general schema resulting from equation (2.32).

If we define value c such that $\int_0^{\varphi} \xi dt' = \log(c^{-1})$ then we have that the equation can be written as

$$\int_0^{\tau-\varphi} a_0(t')dt' = \log({r_2}^{-1}) - \log(c^{-1})$$

which means

$$\int_0^{\tau-\varphi} a_0(t')dt' = \log(cr_2^{-1})\,. \tag{2.35}$$

Finally, by noting that by construction it must be that $c \geq r_2$, then $cr_2^{-1} \in [0,1]$, hence is $(cr_2^{-1}) \sim U[0,1]$. The proofs come from the analytical definition of equations (2.34) and (2.35). □

Some considerations are worth discussing. Firstly, any algorithm which has to solve an equation of the form of equation (2.32) can use the results of equations (2.34) and (2.35). Namely the fact that, if sampling a value $\mathcal{E}xp\,(\xi)$ a value smaller than φ is obtained, then the system does not change up to time $t+\tau$, and the algorithm can perform some operations (i.e. change the state of the system) at time $t+\tau$ accordingly to other probability functions. So for instance in the SSA the next reaction to fire was chosen accordingly to weighted probability. Differently, if sampling a value $\mathcal{E}xp\,(\xi)$ we get a value greater than φ, then a proper choice is to increase the system clock to value $t+\varphi$, change the system state to be some state $\mathbf{x}'$, which can be determined only when we contextualize these functions, and re-start the algorithmic iteration. Indeed, this implies re-sampling a new value for τ which is now $\mathcal{E}xp\,(a_0(\varphi))$. Such a general schema is represented in FIGURE 2.5.

Secondly, as expected equation (2.32) directly relates to the SSA. In fact, in the SSA no value φ can be determined since $\varphi \to +\infty$ and in fact such an equation, once that the case $\tau \geq \varphi$ is disregarded, is equivalent to equation (2.26) in the SSA.

Thirdly, a general consideration on the assumptions to derive this theorem is worth discussing. The target function $p(\tau \mid \mathbf{x}, t)$ denotes the probability of the next event to occur at time τ but what this event actually is not specified, and hence the undetermined state $\mathbf{x}'$ in the formula. In fact, the only consideration which holds for any possible system, is that such equation denotes the probability that the system does not change in the time window $[t, t+\tau)$. The generality of this result makes it suitable for proving the correctness of SSA-based algorithms for the stochastic simulation of biological systems with delays.

2.5 Transition Systems and Bisimulations

In this section we recall basic notions of formal languages theory needed in the second part of the thesis. We recall the classical definitions of *Labeled Transition System* (LTS), *bisimulation* relation over LTSs, and we show how a LTS can be specified by means of inference rules. Moreover, we introduce a format for the inference rules which guarantees an important property of bisimulation.

A LTS is a mathematical model describing something having a notion of state, or configuration, which may evolve by performing steps describing changes in the state. A LTS is formally defined as follows.

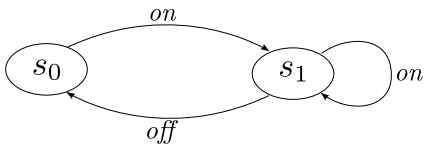

FIGURE 2.6: An example Labelled Transition System.

Definition (Labelled Transition System) A *Labelled Transition System* (LTS) is a triple $(\mathcal{Q}, \mathcal{A}, \mathcal{T})$ consisting of

- a set $\mathcal{Q}$ of states;
- a ternary relation $\mathcal{T} \subseteq (\mathcal{Q} \times \mathcal{A} \times \mathcal{Q})$, known as a transition relation.

If $(q, \alpha, q') \in \mathcal{T}$ we write $q \xrightarrow{\alpha} q'$ and we say that q performs a *transition* becoming q' and *exhibiting* label α. A LTS is finite if $\mathcal{Q}$ is finite, and a LTS is *finite branching* if the set $\{(q, \alpha, q') \in \mathcal{T}\}$ is finite, for any possible q. In some of the cases we consider in this thesis the LTSs we use can be either finite or not, however they are always finite branching.

In a LTS, a state q is reachable from another one q_0 if a system in state q_0 can perform a finite, and possibly empty, sequence of transition at the end of which the state of the system is q. More precisely, q is reachable from q_0 if either $q_0 = q$, or there exist $q_1, \ldots, q_n \in \mathcal{Q}$ and $\alpha_1, \ldots, \alpha_{n+1} \in \mathcal{A}$ such that

$$q_0 \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} q_n \xrightarrow{\alpha_{n+1}} q\,.$$

In a LTS the label of a transition usually denotes the event that has caused the transition. Often, the set of labels of a LTS contains a special label denoting an hidden action, which is typically denoted as τ and represents an internal action of a system, which is not observable from the external.

As an example, in FIGURE 2.6 a LTS describing the behavior of a simple timed switch is given, circles represent states s_0 and s_1. The state of the system in which the switch is off is represented by s_0, and s_1 represents the state in which the switch is on. Initially the switch is off, hence the system initial state is s_0. Arrows represent transitions modeling the interaction with the switch, with the labels describing actions performed. Once the switch is turned on, the system moves from s_0 to s_1 if it is off. After some time the switch is on, it switches off automatically, moving to s_0. If we turn on the switch when it is already working, it simply remains on. These transitions are $s_0 \xrightarrow{on} s_1$, $s_1 \xrightarrow{off} s_0$ and $s_1 \xrightarrow{on} s_1$.

LTSs may describe the behavior of the modeled system in great detail. Relations on states of a LTS can be defined to compare the behavior of two modeled systems. There are many reasons why it is important to equate systems. For instance, checking that a particular system satisfies a specification reduces to checking equivalence of the LTS modeling the system and the LTS describing the specification. Moreover, once that two systems are equivalent, under some conditions one of them can replace the other as part of a bigger system. The behavioral equivalences we consider are reflexive, transitive and symmetric relations that relate systems that are not distinguished by any external observer, according to a given notion of observation. For a broad introduction to such a topic the reader is referred to (van Glabbeek, 1990a; van Glabbeek, 1993).

We recall here the notion of *strong bisimulation* equivalence (Milner, 1980; Park, 1981) which relates two states in a LTS when they are step by step able to perform transitions with the same labels.

Definition (Strong Bisimulation) Given a LTS $(\mathcal{Q}, \mathcal{A}, \mathcal{T})$ a binary relation $R \subseteq \mathcal{Q} \times \mathcal{Q}$ is a *bisimulation* if, for any $(p, q) \in R$, the following propositions hold:

$$\forall p \in \mathcal{Q}, \alpha \in \mathcal{A}.\, p \xrightarrow{\alpha} p' \implies \exists q' \in \mathcal{Q}.\, q \xrightarrow{\alpha} q' \wedge (p', q') \in R$$
$$\forall q' \in \mathcal{Q}, \alpha \in \mathcal{A}.\, q \xrightarrow{\alpha} q' \implies \exists p' \in \mathcal{Q}.\, p \xrightarrow{\alpha} p' \wedge (q', p') \in R$$

It is possible to show that a bisimulation is reflexive, transitive and symmetric and that

$$\sim = \bigcup \{R \mid R \text{ is a bisimulation}\}$$

is a bisimulation and is the largest among all. Bisimulation can intuitively be seen as a two players game. Each player moves accordingly to rules given by the labels of the LTS. Each time *any* of the two player moves, the other should make the same move of the opposer. The players are bisimulation-equivalent, and they are said to be *bisimilar*, if none of them is capable to find a sequence of moves to defeat his opponent. Although weaker notions of bisimulation exist such as weak bisimulation considering *hidden* moves, in the rest of the thesis we consider only the strong bisimulation.

Besides the mathematical importance of bisimulation investigating whether it is a useful equivalence notion in the context of biological systems is still research topic. In fact, such a relation is defined accordingly only to the LTSs, the mathematical structure we consider independently on the context of applications of our theories. Actually, such relation does not seem to play a crucial role in the context of modeling biological systems. Indeed, it would be more helpful to have biologically inspired notions of equivalences.

One of the properties required to a bisimulation is to be a congruence, to this extent let us recall some preliminary notions. Let us consider a countably infinite set of *variables* V, ranged over by x, y, z, ... A *signature* consists of a set of *function symbols*, disjoint from V, together with an *arity* mapping that assigns a natural number $ar(f)$ to each function symbol f. Functions of arity zero are usually called *constants*, while function of arity greater than zero are usually called *operators*. Given a constant f we write f for $f()$.

We introduce now the notion of *open terms* over a signature.

Definition (Open terms) The set of *open terms* $T(\Sigma)$ over a signature Σ is the least set such that: (i) $V \subseteq T(\Sigma)$, and (ii) given a function symbol f and t_1 , ... , $t_{ar(f)} \in T(\Sigma)$ it holds $f(t_1, \ldots, t_{ar(f)}) \in T(\Sigma)$. The set $T(\Sigma)$ is ranged over by t, u, v, ...

Terms that does not contain variables are usually called *closed terms*. The set of closed terms is denoted by $Tg(\Sigma)$. The set of closed terms over Σ gives the *term algebra* of Σ. We recall that, given a signature Σ, a Σ-*algebra* is a pair (A, Σ_A), where A is a set called *carrier* and Σ_A is a set of functions $\{f_A : A^n \mapsto A \mid f \in \Sigma, ar(f) = n\}$. Essentially, (A, Σ_A) is an interpretation of Σ. Now, the term algebra of Σ is the Σ-algebra having $Tg(\Sigma)$ as carrier, and, for each $f \in \Sigma$ with $ar(f) = n$, a function mapping closed terms t_1, ..., t_n to term $f(t_1, \ldots, t_n)$.

A *substitution* is a mapping $\eta : V \mapsto T(\Sigma)$. A substitution can be extended trivially to a mapping from terms to terms, namely, $\eta(t)$ is the term obtained by replacing all the variables occurring in t by $\eta(x)$. A substitution is called instantiation if it maps variables to closed terms. A *context* $C[x_1, \ldots, x_n]$ denotes an open term in which at most the distinct variables $x_1, \ldots, x_n$ may appear. The term $C[t_1, \ldots, t_n]$ is obtained by replacing all occurrences of variables x_i in $C[x_1, \ldots, x_n]$ by t_i, for $1 \leq i \leq n$.

Once we introduced these notions, we can recall the definition of congruence for a generic equivalence relation.

Definition (Congruence) Assume a signature Σ. An equivalence relation R over $T(\Sigma)$ is a *congruence* if, for all $f \in \Sigma$, it holds

$$\forall i = 1, \ldots, ar(f). (t_i, u_i) \in R, \implies (f(t_1, \ldots, t_{ar(f)}), f(u_1, \ldots, u_{ar(f)})) \in R\,.$$

In this sense, having a bisimulation which is a congruence means that, for any function in the signature, the composition with such a function preserve the bisimilarity property, which is an important feature for process algebras relying on compositionality.

Following the *Structural Operational Semantics* (SOS) approach (Plotkin, 1981; Plotkin, 2004), LTSs in which states are terms built over some signature are usually specified by means of a set

of *inference rules*. An inference rule for the specification of a LTS, termed a transition rule, is a logical rule having the form

$$\frac{t_1 \xrightarrow{l_1} t'_1 \quad \cdots \quad t_n \xrightarrow{l_n} t'_n}{t \xrightarrow{l} t'}$$

where $t_i \xrightarrow{l_i} t'_i$, $1 \leq i \leq n$ are the *premises* and $t \xrightarrow{l} t'$ the *conclusion*. A transition rule states that whenever the premises are transitions of the LTS, then also the conclusion is a transition of the LTS. Side conditions can be associated to a transition rule with the effect of imposing that the conclusion of the rule is a transition of the LTS whenever both the premises and the side conditions are satisÞed. A transition rule without premises is called an axiom, and a non empty and possibly infinite LTS can be specified by providing a set of transition rules with at least one axiom. We assume the standard way to assign a LTS to a set of inference rules (Aceto *et al.*, 2001).

We recall here a special format of inference rules, the *De Simone* format. A De Simone language (De Simone, 1984; De Simone, 1985) consists of a signature together with a set of inference rules, extended with transition rules for recursion. Most of the languages we cited in SECTION 1.1 are De Simone languages. The De Simone format is as follows (Vaandrager, 1993).

Definition (De Simone Format) Assume a signature Σ. An inference rule is in the *De Simone format* if it has the form

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\}}{f(x_1, \ldots, x_{ar(f)}) \xrightarrow{a} t}$$

where $I \subseteq \{1, \ldots, ar(f)\}$, x_i and y_i are all distinct and are the only variables occurring in the inference rule. Moreover, t does not contain variables x_i for $i \in I$ and has no multiple occurrences of variables.

Such a format guarantees a useful property for LTSs and bisimulations.

PROPOSITION 2.5.1. *Assume a signature Σ. A* LTS *built from a set of inference rules in the De Simone format on Σ has a strong bisimulation which is a congruence.*

For a proof of this and for a broad introduction to SOS the reader is referred to (Aceto *et al.*, 2001). Among all the possible variants of the SOS, in this thesis we mainly focus on the Starting Terminating approach (van Glabbeek & Vaandrager, 1987; Hennessy, 1988; van Glabbeek, 1990; Bravetti *et al.*, 1998; Bravetti & Gorrieri, 2002) which turns out to be suitable to model systems with delays, namely systems in which actions are not instantaneous, but described by two detached starting and completion transitions.

Chapter 3

Example applications

In this chapter we present some classes of biological systems which is reasonable to model by means of delays: *cellular* models, *epidemic* models and *evolutionary* models. For each of the target systems we try to identify simple models, either deterministic or stochastic, and the role of the delays in deploying the models.

At the end of the chapter, we build a deterministic tumor growth model in which we show the use of delays to abstract phase-passage in the cell cycle. We perform numerical simulations of the DDEs and discuss the result.

3.1 Target systems

In the next section we present some simple models with delays either specified by means of differential equations or in a reaction-style notation. For each of the model we present we try to give enough intuition about the biology behind the model. The models we present are not fully analyzed in this thesis, but references to works where models are extensively analyzed are given. For an exhaustive introduction of the models we present the reader is referred to (Murray, 1989).

The types of models we discuss are chosen with a specific motivation: they are the core of most complex models built on top of those. This is a quite general phenomenon in a compositional approach where complex models embed simpler ones. We recall that, since ODEs are not compositional, the solutions of complex models can not be obtained by composing the solutions of simpler ones.

As an example of such models, most of complex modern *epidemics* or *evolutionary models* are based on the original models of (Kermack & McKendrick, 1927) and (Lotka, 1920; Volterra, 1926), respectively. And this kind of observation can be done for quite huge classes of biological systems. The aim of the next sections is, for some candidate target systems, define simple models with delays as extensions of their corresponding classic non-delayed models.

In the next, we assume a reaction to be represented in notation

$$R \overset{k,\sigma}{\longmapsto} P$$

where R, P and k have the usual meaning, the new parameter $\sigma \in \mathbb{R}$ represents the delay of the reaction.

3.1.1 Cellular Models

In this section we define a model of simple gene regulation, where the protein product from translation controls transcription. We start contextualizing this model.

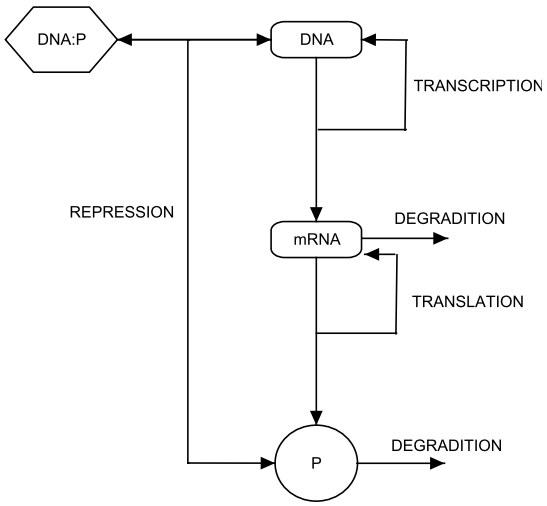

FIGURE 3.1: A gene regulatory model: transcription, translation and repression.

Biological preamble. Some organisms, such as most bacteria, consist of a single cell, other organisms, such as humans, are multicellular. Cells contain all the genetic material defining an organism. Two different kinds of genetic material exist: *deoxyribonucleic acid* (DNA) and *ribonucleic acid* (RNA). DNA is a double-stranded molecule which carries the genetic information of a cell and consists of thousands of *genes*. RNA is chemically similar to DNA excepts for some molecules constituting these macromolecules, RNA is a single-stranded molecule. Genes serve as an instruction set on how to build a *protein* molecule. Proteins, DNA and RNA constitute the most important macromolecules of living organisms.

Proteins perform crucial tasks for the cell functions or serve as building blocks for more complex biological structures. The flow of information from the genes determines the protein composition and the structure of a protein determines its functionalities. Hence, this process of protein creation is crucial for the functions of a cell.

The DNA is situated in the *nucleus*, the most internal area of a cell, and is organized into vector-style biological structures called *chromosomes*. When proteins are needed, the corresponding genes are *transcribed* into RNA via a process named *transcription*. The RNA is then processed so that some of its parts, which do not contain useful coding information, are removed becoming, in a very simplistic approximation, *messenger RNA* (mRNA). The mRNA is then it is transported out of the nucleus where the proteins are built based upon the code in the mRNA in a process named, *translation*. The protein product from gene expression may bind to a regulatory region on the DNA and repress transcription.

All the processes we describe involve a lot of different molecules, so for instance the reactions leading to creation of mRNA are catalyzed by *enzymes*, *coenzymes*, *cofactors* and a lot other chemical complexes. Since the model we consider are not exhaustive, we omit describing at a higher detail our target systems. From the simple model we create it is possible to create more complex models by considering components omitted here, as done for instance in (Monk, 2003; Bratsun *et al.*, 2005; Barrio *et al.*, 2006; Momiji & Monk, 2008).

Model construction. We start by considering DNA molecules in the system at time t denoted as $DNA(t)$, mRNA molecule as $mRNA(t)$ and a generic protein as $P(t)$. The complex composed by the protein bounded to the DNA strand is denoted as $DNA{:}P$. In FIGURE 3.1 the target system is graphically represented.

For the sake of shortening the deterministic equations we present in the next, we shorten the acronyms of the molecules involved to D (DNA), R ($mRNA$) and $D_P(DNA{:}P)$. In this system

we model the following events:

- a DNA molecule *transcripts* a molecule of $mRNA$ at rate α, without consuming;
- an $mRNA$ molecule *translates* a protein P at rate β, without consuming;
- a molecule of DNA and a protein can freely bind and unbind at rates γ_1 and γ_2, respectively;
- both the $mRNA$ and the proteins degrade at rate δ_1 and δ_2, respectively .

We can define a DDEs model with one equation for each population as follows

$$\frac{dD}{dt} = -\gamma_1 DP + \gamma_2 D_P \qquad \frac{dR}{dt} = \alpha D(t-\sigma_1) - \delta_1 R$$
$$\frac{dD_P}{dt} = \gamma_1 DP - \gamma_2 D_P \qquad \frac{dP}{dt} = \beta R(t-\sigma_2) - \gamma_1 DP + \gamma_2 D_P - \delta_2 P$$

Here, D, R, D_P and P stand for the populations at time t: term $\gamma_1 DP$ models the binding of DNA and protein, term $\gamma_2 D_P$ models the corresponding unbinding, $\delta_1 R$ and $\delta_2 P$ model the linear degradation of mRNA and protein. Differently, terms $\alpha D(t-\sigma_1)$ and $\beta R(t-\sigma_2)$ model transcription and translation of mRNA and protein, respectively. The former linear process is assumed to have duration σ_1, namely new molecules of mRNA at time t are the result of transcriptions started at time $t-\sigma_1$ and at time t mRNA molecules become available. The latter process is is assumed to have duration σ_2, namely the time to translate a protein P from a molecule of mRNA is σ_2 and hence new proteins at time t are the result of translations started at time $t-\sigma_2$. In this model, the use of delays is crucial to abstract all the steps of transcription and translation once that all the involved enzymes and other general molecules are abstracted to keep the model reasonably small. More precisely, in the case of transcription by using delays we can abstract the binding of RNA polymerase and cofactors with a DNA molecule, the move of RNA polymerase along the DNA and the combination of the nucleotides to create mRNA. In the case of translation by using delays we abstracted the move of the mRNA from the nucleus to the cytoplasm, its binding with ribosomes, the move of the ribosomes along the mRNA and creation of proteins with the help of transfer RNA bound to amino acids.

The equivalent reaction-style model which describes the time evolution the population composed by DNA, $mRNA$, $DNA{:}P$ and P once the populations are discrete, is given by the following reactions

$$(R_1)\ DNA \overset{\alpha,\sigma_1}{\longmapsto} DNA + mRNA \qquad (R_2)\ mRNA \overset{\beta,\sigma_2}{\longmapsto} mRNA + P$$
$$(R_3)\ DNA + P \overset{\gamma_1,0}{\longmapsto} DNA{:}P \qquad (R_4)\ DNA{:}P \overset{\gamma_2,0}{\longmapsto} DNA + P$$
$$(R_5)\ mRNA \overset{\delta_1,0}{\longmapsto} \emptyset \qquad (R_6)\ P \overset{\delta_2,0}{\longmapsto} \emptyset\,.$$

Clearly, reaction R_3 models binding of DNA and the protein, reaction R_4 the unbinding, reactions R_5 and R_6 model the degradation of mRNA and protein, all these reactions are non-delayed. Reactions R_1 and R_2 have delay σ_1 and σ_2, respectively: the former models the delayed transcription and the latter the delayed translation. The propensity functions of these reactions are defined by the following analytical expressions

$$a_1 = \alpha[DNA] \qquad a_2 = \beta[mRNA]$$
$$a_3 = \gamma_1[DNA][P] \qquad a_4 = \gamma_2[DNA{:}P]$$
$$a_5 = \delta_1[mRNA] \qquad a_6 = \delta_2[P]\,.$$

By means of the analysis techniques for stochastic models with delays we present in the next chapters this model can be analyzed and compared to the numerical solution of the deterministic counterpart.

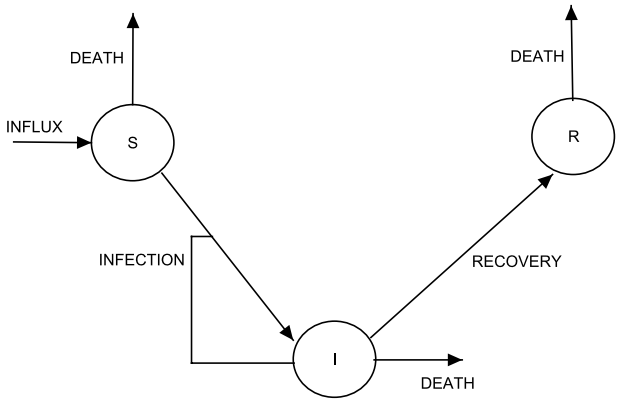

FIGURE 3.2: A Susceptible-Infectious-Recovered epidemics model.

3.1.2 Epidemic Models

In this section we present a simple Susceptible-Infectious-Recovered (SIR) model, one of many possible *epidemic* models. Differently from the gene expression model, due to its generality this model can be understood without any special biological background.

SIR models, originally defined in (Kermack & McKendrick, 1927), deals with populations and illnesses within the populations. More precisely, an SIR model is an epidemiological model that computes the theoretical number of people infected with a contagious illness in a closed population over time. The acronym of these models derives from the fact that they involve coupled equations relating the number of susceptible people, the number of people infected and the number of people who have recovered. As intuitive, susceptible people are not affected by the illness, infected have contracted the illness once they where susceptible, and recovered have recovered from infection and can not get infected anymore (i.e. infection confers permanent immunity).

As expected, this kind of models are very general and nowadays complex predictive immunology models (Beretta *et al.*, 2002; D'Onofrio *et al.*, 2007; Zhanga *et al.*, 2008) have been defined with much more complex features: spatiality information, illness-related information diffusion, vaccinations recurrent illnesses, non-instantaneous recovery. In FIGURE 3.2 the target system is graphically represented.

Model construction. Let $S(t)$, $I(t)$ and $R(t)$ denote the susceptible, the infected and the recovered people in the population at time t. Let us assume that the events which can happen in the system are the following:

- new people enter the population at constant rate α;
- susceptible and recovered die at rate δ_1;
- infected die at rate δ_2;
- a susceptible gets infected at rate β;
- an infected gets recovered at rate γ;

We can define a DDEs model with one equation for each population as follows

$$\begin{aligned}
\frac{dS}{dt} &= \alpha - \delta_1 S - \beta S(t-\sigma_1) I(t-\sigma_1) \\
\frac{dI}{dt} &= \beta S(t-\sigma_1) I(t-\sigma_1) - \gamma I(t-\sigma_2) - \delta_2 I \\
\frac{dR}{dt} &= \gamma I(t-\sigma_2) - \delta R\,.
\end{aligned}$$

Here, S, I and R stand for the populations at time t: term α models constant influx of new susceptible and $\delta_1 S$, δR and $\delta_2 I$ the linear death. Differently, terms $\beta S(t-\sigma_1)I(t-\sigma_1)$ and $\gamma I(t-\sigma_2)$ denote the interaction between the populations modeling the infection and the recovery in the populations. The former non-linear process is assumed to have duration σ_1, namely new infected people at time t were susceptible at time $t-\sigma_1$ and at that time started becoming infected. The latter process is is assumed to have duration σ_2, namely the time to recover by vaccination, for instance, is $\sigma_1$and hence new recovered at time t were infected at time $t-\sigma_2$ and at that time started the recovery process. Original SIR models are based on ODEs hence infection and recovery are instantaneous events however, from the biologically point of view, the use of delays seems more reasonable for the underlying events assumed by these dynamics. The reason for this statement are fairly intuitive.

The equivalent reaction-style model which describes the time evolution the discrete population composed by S, I and R is given by the following reactions

$$
\begin{aligned}
&(R_1)\ \emptyset \xmapsto{\alpha,0} S && (R_2)\ S \xmapsto{\delta_1,0} \emptyset \\
&(R_3)\ R \xmapsto{\delta_1,0} \emptyset && (R_4)\ I \xmapsto{\delta_2,0} \emptyset \\
&(R_5)\ S+I \xmapsto{\beta,\sigma_1} 2I && (R_6)\ I \xmapsto{\gamma,\sigma_2} R
\end{aligned}
$$

Clearly, reaction R_1 models the influx of susceptible, reactions R_2, R_3 and R_4 model the population death, all these reactions are non-delayed. Reactions R_5 and R_6 have delay σ_1 and σ_2, respectively: the former models the delayed infection effect and the latter the delayed recovery effect. The propensity functions of these reactions are defined by the following analytical expressions

$$
\begin{aligned}
&a_1 = \alpha && a_2 = \delta_1[S] \\
&a_3 = \delta_1[R] && a_4 = \delta_2[I] \\
&a_5 = \beta[S][I] && a_6 = \gamma[I]\,.
\end{aligned}
$$

By means of the analysis techniques for stochastic models with delays which we will present in the next chapters of the thesis it will be clear how such model can be analyzed and potentially compared to the numerical solution of the deterministic counterpart.

3.1.3 Evolutionary Models

Originally, *prey-predator* models, also known as Lotka-Volterra models (Lotka, 1920; Volterra, 1926) have been studied for describing the dynamics of competitive populations living in a same environment, historically rabbits and wolves. As for the SIR model, the generality of the prey-predator model we describe is such that it can be understood without any special biological background.

These types of models play a crucial role in bio-economics, namely the management of renewable resources, and are based upon the competition between the involved species together with their evolution simply for the purpose of seeking resources to sustain their struggle for their existence. Depending on their specific settings of applications, these models can be interpreted as resource-consumer, parasite-host, virus immune-system interactions or even as special SIR models. They deal with the general loss-win interactions and hence may have applications outside of ecosystems. A lot of variants of such models exist which differ in the deployed features: for instance some assume bounded populations size, others model harvesting either of preys or predators. Generally, more complex models as (Tyson, 1973; Marti & Ruan, 2001; Ruan, 2009; Caravagna *et al.*, 2010) have been defined by staring from basic Lotka-Volterra equations.

Model construction. Let $X(t)$ and $Y(t)$ denote the preys and predators populations at time t. In FIGURE 3.3 our target system is graphically represented.

Let us assume that the events which can happen in the system are the following:

- preys reproduce and die at rates α and $\alpha\gamma^{-1}$, respectively;

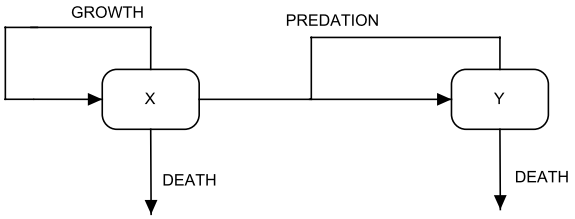

FIGURE 3.3: A prey-predator model.

- predators eat preys at rate β;
- predators die at rate δ.

We can define a DDEs model with one equation for each population as follows

$$\frac{dX}{dt} = \alpha X\left(1 - \frac{X}{\gamma}\right) - \beta X(t-\sigma)Y(t-\sigma)$$
$$\frac{dY}{dt} = \beta X(t-\sigma)Y(t-\sigma) - \delta Y\,.$$

Here, X and Y stand for the populations at time t: term $\alpha X(1 - X/\gamma)$ models the *logistic growth with plateau* γ of the preys and term δY the death of the predators. Differently, term $X(t - \sigma)Y(t - \sigma)$ denotes the non-linear interaction between the populations modeling the eating of preys by predators. Such process is assumed to increase (resp. decrease) the population size of predators (resp. preys), a delay is used to model the fact that the change rate of predators depends on the number of prey and predators at some previous time $t - \sigma$. From the biologically point of view such a delay is justified by the following considerations: feeding is related to hunting and in this model predators reproduce once they feed, of course reproduction is not, with respect to feeding, an instantaneous process. As a consequence, the new predators entering the population at time t are assumed to be the result of a reproduction started at time $t - \sigma$, if σ is the average gestation time for the predators population. Notice that original models are non-delayed and as a consequence it is as they assume that reproduction is an instantaneous event in the system. Again the use of delays seems to be reasonable also in this models.

Summarizing, this model represents a prey-predator model with delayed predation effect. We present now an equivalent definition of such model in a reaction-style notation. First of all, let us denote with X and Y the two species, assuming now discrete values, of preys and predators, accordingly to the events we want to model and by rewriting the logistic equation as $\alpha X - \alpha X^2/\gamma$, we can write the equivalent 4 reactions

$$(R_1)\ X \overset{\alpha,0}{\longmapsto} 2X \qquad (R_2)\ 2X \overset{\overline{\alpha},0}{\longmapsto} X$$
$$(R_3)\ X + Y \overset{\beta,\sigma}{\longmapsto} 2Y \qquad (R_4)\ Y \overset{\delta,0}{\longmapsto} \emptyset$$

where $\overline{\alpha} = \alpha\gamma^{-1}$. Clearly, reactions R_1 and R_2 model the growth/death of preys, and reaction R_4 the death of the predators, all these reactions are non-delayed. Reaction R_3 has delay σ and models the delayed effect of predation and reaction. The propensity functions of these reactions are defined by the following analytical expressions

$$a_1 = \alpha[X] \qquad a_2 = \overline{\alpha}[X](X-1)/2$$
$$a_3 = \beta[X][Y] \qquad a_4 = \delta[Y]\,.$$

By means of the analysis techniques for stochastic models with delays which we will present in the next chapters of the thesis it will be clear how such model can be analyzed and potentially compared to the numerical solution of the deterministic counterpart.

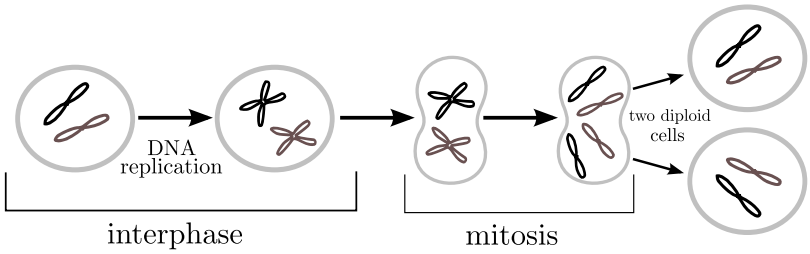

FIGURE 3.4: The cell cycle: interphase and mitotic phase.

3.2 A deterministic model of the cell cycle

In this section, we discuss a deterministic model of the cell cycle with delays as an abstraction of a corresponding non-delayed model. Indeed we start creating a complete ODE model of the cell cycle and, secondly, we try to reduce model size by means of a delay, leading to the definition of a DDE model. Such a model is then analyzed by performing numerical simulations of the DDEs.

Biological preamble. As we said in the discussion of the cellular model of SECTION 3.1.1, cells are at the base of life. Obviously, cells are likely to grow, die and replicate. Cell death is either auto-induced or injury-induced. When cell death is programmed it can be via either *apoptosis* or *autophagy*. Cell replication is a process involving cell growth and characterized by a deeply studied biological mechanism, the *cell cycle*. The cell cycle is a series of events that culminates in the asexual reproduction of a cell via cell division.

In a typical cell cycle, the parent cell doubles its volume, mass, and complement of chromosomes, then sorts its doubled contents to opposite sides of the cell, and finally divides in half to yield two genetically identical offspring. This is a cycle since the parent cell backs to its original size and chromosome number, and begins another cell cycle. If the organism we consider is unicellular no differentiation is possible for daughter cell, differentiation is instead possible in multicellular organisms. Differentiating cells may differ from their parent cell and from each other in terms of size, shape, and differentiation state. Dependently on the type of cell the time required for completion of the cell cycle varies. So for instance embryonic cells may complete a cycle in around 8 minutes, whereas somatic cells in around 10-24 hours.

The cell cycle, for some type of cells, consists of four phases: gap phase 1 (G_1), synthesis (S), and gap phase 2 (G_2). Interphase is followed by mitosis (M) with nuclear division and cytokinesis, namely cell division. The first three phases (G_1, S, G_2) are called *interphase* and accounts almost 90% of cell cycle time, the fourth phase is called *mitosis*.

Phase G_1 lasts from the end of mitosis to the beginning of S phase. During this phase, the cell chooses either to replicate its DNA or to stop for a period the cell cycle. In phase S chromosomes are replicated. In phase G_2 cells can exit the cell cycle as in G_1 phase. Finally, mitosis is divided into five internal stages leading to creation of a new daughter cell starting a new cell cycle.

A graphical representation of the cell cycle is given in FIGURE 3.4.

A complete model with only phase-passages. In the literature non-delayed models of the cell cycle as been studied (Chen *et al.*, 2004; Li *et al.*, 2004; Csikasz-Nagy *et al.*, 2006), in this section we concentrate on a very model with only phase passages.

A complete definition of the cell cycle consists of 4 populations: the cells in the phase G_1 at time t, $T_{G_1}(t)$, the cells in the phase S at time t, $T_S(t)$, the cells in the phase G_2 at time t, $T_{G_2}(t)$, and the cells in the mitotic phase at time t, $T_M(t)$. The events to model are the passages from each of the phases to the next one, with a cyclic passage from the mitotic phase to phase G_1 as depicted in FIGURE 3.5.

Hence, by denoting the rate of passage from G_1 to S as α_1, the one from S to G_2 as α_2, the one from G_2 to M as α_3 and the one from M to G_1 as α_4, the following ODEs can be used as a

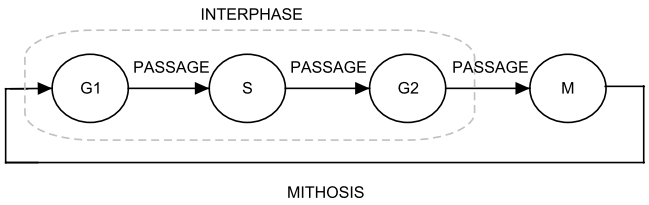

FIGURE 3.5: A complete model of the cell cycle.

basic model for the passage of phases of the cells:

$$\frac{dT_{G_1}}{dt} = -\alpha_1 T_{G_1} + 2\alpha_4 T_M \qquad \frac{dT_S}{dt} = \alpha_1 T_{G_1} - \alpha_2 T_S$$
$$\frac{dT_{G_2}}{dt} = \alpha_2 T_S - \alpha_3 T_{G_2} \qquad \frac{dT_M}{dt} = \alpha_3 T_{G_2} - \alpha_4 T_M \,.$$

Notice that the last passage from a single cell in mitotic phase to two new cells in phase G_1 is modeled by term $2\alpha_4 T_M$ in the equation for T_{G_1}. Of course, this model is still not complete since it should be enriched with other terms modeling the events which can happen to a cell in any specific phase of the cell cycle (e.g. cell death). The reaction-style representation of this model is given by the following set of reactions

$$T_{G_1} \overset{\alpha_1}{\longmapsto} T_{G_2} \qquad T_{G_2} \overset{\alpha_2}{\longmapsto} T_S$$
$$T_S \overset{\alpha_3}{\longmapsto} T_M \qquad T_M \overset{\alpha_4}{\longmapsto} 2T_{G_1}$$

where all the reactions correspond the terms appearing in the ODE model. Although this model is very simple, it requires to know the values of parameters as the rate of passage from each phase to the next one. Sometimes this information is not available (i.e. experimental reasons may not permit to measure these quantities) and this model can not be analyzed numerically. Notice that also the reactions-based model is likely not to be analyzed since it depends on the same parameters. If this is the case, we can either try to analytically analyze the ODEs or try to switch to another representation of the same model. More precisely, by analyzing the model description we can discover that there is one information which we did not use to build the ODEs, namely the fact that the average length of the cell cycle for certain type of cells and under some conditions is known. This information can be used to build a smaller version of this model, we discuss now how this can be done by using a delay.

A model with a delay in phase-passages and cell death. In (Villasana & Radunskaya, 2003) is introduced a DDEs model of tumor growth that includes the immune system response and a phase-specific drug able to alter the natural course of action of the cell cycle of the tumor cells. The model in (Villasana & Radunskaya, 2003) considers three populations of cells: the immune system, the population of tumor cells during cell cycle interphase, and the population of tumor cells during mitosis. A delay is used to model the duration of the interphase, hence the model includes a delayed event that is the passage of a tumor cell from the population of those in the interphase to the population of those in the mitotic phase. In the model the effect of a phase-specific drug, able to arrest tumor cells during the mitosis, is studied. Such a drug has a negative influence also on the survival of cells of the immune system.

Here we study a simplified version of the model, presented in SECTION 4.1.2 of (Villasana & Radunskaya, 2003), where the effects of the immune response and of the drug are not taken into account. The simplified model considers only tumor cells classified in two populations such that:

- $T_I(t)$ denotes the population of tumor cells during interphase at time t;

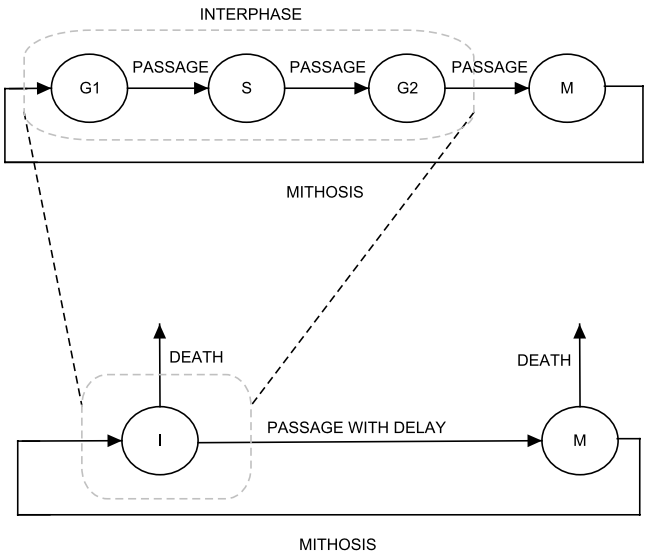

FIGURE 3.6: A model of the cell cycle with a delay.

- $T_M(t)$ denotes the population of tumor cells during mitotic phase at time t.

Practically, from the original ODEs model the population abstracted are all the ones described by the equations for $T_{G_1}(t)$, $T_S(t)$ and $T_{G_2}(t)$ which now are collapsed in the unique population $T_I(t)$.

As an abstraction of the ODEs model just presented, the cell cycle with respect to these two populations can be described by using the following DDEs:

$$\begin{aligned}\frac{dT_I}{dt} &= -a_1 T_I(t-\sigma) + 2a_4 T_M \\ \frac{dT_M}{dt} &= a_1 T_I(t-\sigma) - a_4 T_M\end{aligned}$$

where the term $a_1 T_I(t-\sigma)$ models the passage of a cell from the interphase to the mitotic phase with rate a_1 and with delay σ. All the other terms have the same meaning as in the ODEs model, since they are valid ODEs terms. The delay here represents the information not appearing in the ODEs model, namely the average length of the cell cycle seen as the passage of a cell through all the phases. This model is smaller than the one composed by the ODEs and, as a consequence, it requires less parameters.

When such a DDEs model is enriched with other terms modeling events which could happen to a generic cell in the interphase and a cell in the mitotic phase, we get the model presented in (Villasana & Radunskaya, 2003), which is graphically represented in FIGURE 3.6.

This model is mathematically described by the following DDEs:

$$\begin{aligned}\frac{dT_I}{dt} &= 2a_4 T_M - d_2 T_I - a_1 T_I(t-\sigma) \\ \frac{dT_M}{dt} &= a_1 T_I(t-\sigma) - d_3 T_M - a_4 T_M\,.\end{aligned}$$

We discuss now the construction of the DDEs by analyzing the terms appearing in the equations:

- $d_2 T_I$ represents cell death, or apoptosis, for cells in the interphase happening at rate d_2;
- $d_3 T_M$ represents cell death, or apoptosis, for cells in the mitotic phase happening at rate d_3;

	R-I	R-II	R-III / V	R-IV
$\sigma = 0.0$	∞	$(0, 0)$	$(0, 0)$	$(0, 0)$
$\sigma = 1.0$	∞	$(0, 0)$	$(0, 0)$	$(0, 0)$
$\sigma = 10.0$	∞	oscillations to $(0, 0)$	oscillations to ∞	oscillations to $(0, 0)$

TABLE 3.1: The behavior of the DDEs in the regions shown in FIGURE 3.7: divergency is represented by ∞, oscillations are explicitly annotated.

	R-I	R-II	R-III / V	R-IV
a_1	0.6	0.4	1.0	0.8
d	0.6	1.0	1.8	0.8
a_4	0.5	0.5	0.5	0.5
d_2	0.3	0.3	0.3	0.3

TABLE 3.2: The parameters used to simulate the DDEs model.

- $a_4 T_M$ represents the cell mitosis happening at rate a_4 and producing 2 new cells in the interphase (hence the term $2a_4 T_M$ in the equation for T_I);
- $a_1 T_I(t - \sigma)$ models the passage of a cell from the interphase to the mitotic phase with rate a_1 and with delay σ.

In the following we shall denote with d the rate at which mitotic cells disappear, namely $d = d_3 + a_4$. We assume that cells reside in the interphase at least σ units of time; then the number of cells that enter mitosis at time t depends on the number of cells that entered the interphase at least σ units of time before.

By assuming $t_0 = 0$, the delay σ requires the values of T_I and T_M to be given also in the interval $[-\sigma, 0]$ since, in the time window $[0, \sigma]$, the derivatives of the two functions depend on states in such interval. In general, it is possible to define the initial conditions by using two generic functions $\phi_I(t)$ and $\phi_M(t)$ defined in $[-\sigma, 0]$; in (Villasana & Radunskaya, 2003) such functions are assumed to be constant in the considered interval, and equal to the values of T_I and T_M at time 0. For the model we have

$$T_I(t) = T_I(0) \quad T_M(t) = T_M(0) \qquad\qquad \forall t \in [-\sigma, 0]$$

where $T_I(0)$ and $T_M(0)$ model the initial configuration. The analytic study of the DDEs constituting the model gives $(0, 0)$ as unique equilibrium. We performed numerical simulation of the DDEs by using `Mathematica`.

In FIGURE 3.7, taken from (Villasana & Radunskaya, 2003), some results are shown of the study of the model by varying a_1, d and σ and by setting the parameters a_4 and d_2 to 0.5 and 0.3, respectively. Figure 3.7 shows five regions, the results are summarized in TABLE 3.1 and the parameters are listed in TABLE 3.2.

For all the simulation results we are going to present, we always used the same initial state consisting of

$$T_I(0) = T_M(0) = 10^5 .$$

namely 10^5 tumor cells in the interphase and 10^5 tumor cells in mitosis.

When $\sigma = 0$, the region in which the tumor grows is R-I, while in the other regions the tumor decays without showing noticeable oscillations.

When the delay is present ($\sigma > 0$), the growth region is essentially unaltered, but the decay is split into regions in which the tumor has different behaviors: in regions R-II $\cup$ R-IV the tumor still decays, but in regions R-III $\cup$ R-V, when the value of σ is sufficiently large, the equilibrium becomes unstable. This is shown in FIGURE 3.8 and in FIGURE 3.9.

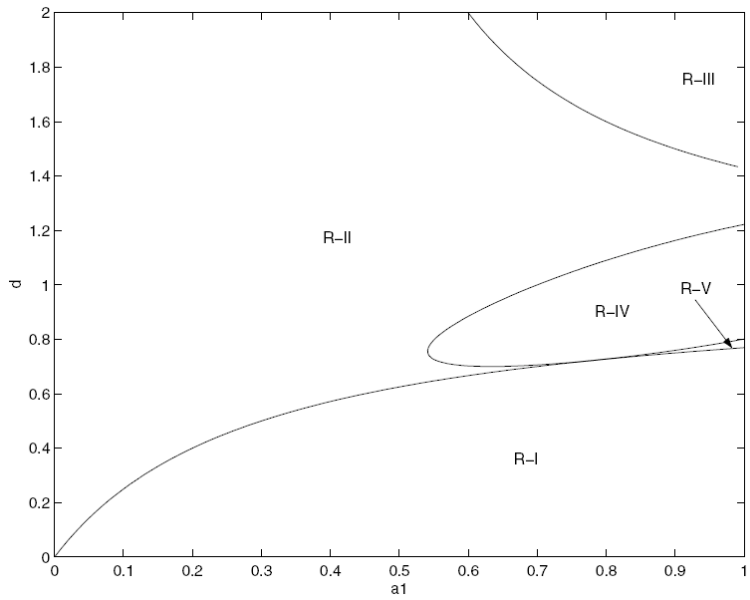

FIGURE 3.7: DDEs tumor growth model and regions of behavior.

	R-I	R-II	R-III / V	R-IV
$\sigma = 1.0$	∞	50	15	238
$\sigma = 10.0$	∞	59	12	440

TABLE 3.3: Value of t^* such that $T_I(t^*) \wedge T_M(t^*) < 1.0$, ∞ means $t^* \to \infty$.

FIGURE 3.8 describes the behavior of the model, obtained by numerical solutions, inside the regions R-I, R-II, R-III, and R-IV, when $\sigma = 1$. In the figure, we can observe that, while the tumor grows in region R-I, it decays in all the other regions.

FIGURE 3.9 describes the behavior of the model when $\sigma = 10$. In regions R-I and R-IV the tumor has the same behavior as before. In region R-II it decays after some oscillations, while in region R-III it expresses an instability around the equilibrium. However, remark that values of T_M and T_I under 0 are not realistic, and, for instance, they could not be obtained by stochastic simulations, which are based on discrete and positive variables.

Indeed, we defined, for every parameter configurations, a property to be observed on the number of cells, namely the first day of simulation in which the eradication of both the populations was visible:

$$T_I(t^*) < 1.0 \ \wedge \ T_M(t^*) < 1.0$$

where

$$\forall t < t^*.\, T_I(t^*) \geq 1.0 \ \wedge \ T_M(t^*) \geq 1.0$$

and, in TABLE 3.3, we summarize the value of t^*. As expected, for any value of σ and parameters in R-I we have that $t^* \to \infty$, but for all the other regions the value for t^* is finite.

Notice that, when for instance in R-II with $\sigma = 10$ we have $t^* = 59$ it is clear from FIGURE 3.9 that even for smaller values of t^* one of the two populations is negative but the other no. Of course, in a stochastic setting with either delayed or non-delayed events, such population would have been considered eradicated and the value of t^* would have been smaller. Such a problem happens only in deterministic frameworks, either DDEs-based or ODEs-based, where the concentrations of the populations are continuous values and is a well-known motivation to stress the importance of stochastic models as previously stated.

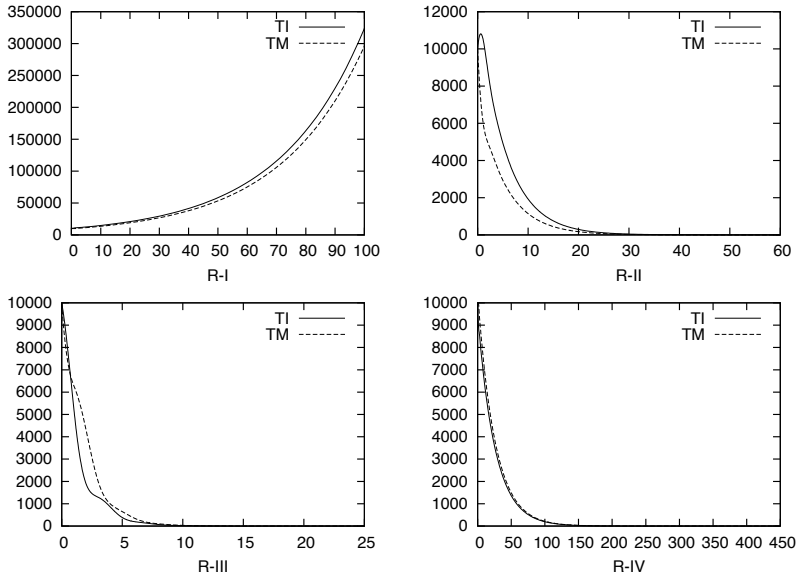

FIGURE 3.8: DDEs numerical approximation ($\sigma = 1$) for the regions of FIGURE 3.7.

We do not discuss into details commenting the biological results shown by this model when analyzed since it has been done in (Villasana & Radunskaya, 2003). However, in the next chapters we use the results outlined in this section to compare DDEs with further analysis techniques based on stochastic simulations presented in the first part of this thesis.

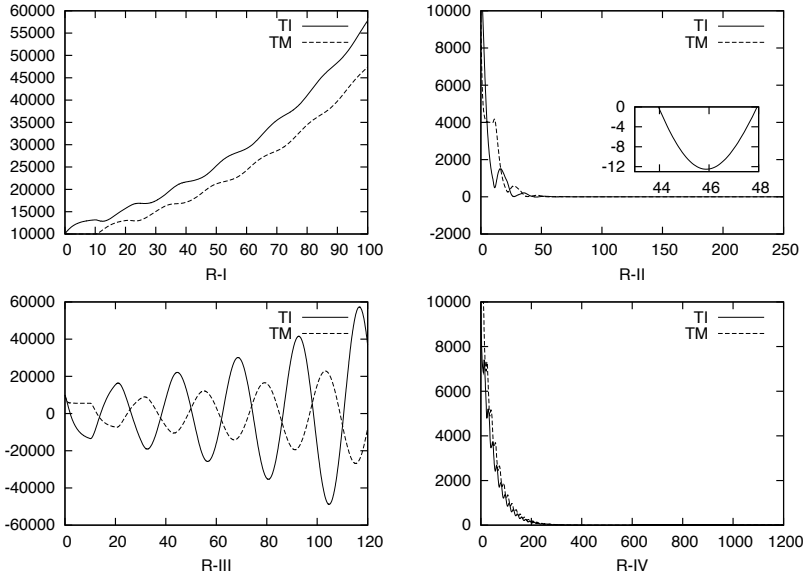

FIGURE 3.9: DDEs numerical approximation ($\sigma = 10$) for the regions of FIGURE 3.7.

Part I

Delay Stochastic Simulation Algorithms (DSSAs)

Chapter 4

Delays as durations

In this chapter we discuss an algorithm for the stochastic simulation of biological systems with delays based on the idea of delays as durations; such an algorithm initially appeared in (Barrio *et al.*, 2006). The crucial idea in this algorithm is that of assigning a delay to a reaction so that every time the reaction starts, it completes after a duration equal to the delay; between the start and the completion of the reaction, the involved reactants are *"locked"*, in the sense that they can not participate in other events involving the system.

Firstly, we give some intuition on this algorithm and, secondly, we present its formal definition; we investigate its correctness by firstly defining a master equation for systems ruled by delays as durations, and then we investigate the mathematical foundation of the algorithm, proving its correctness. At the end of the chapter, we apply the algorithm to the analysis of a stochastic model of the cell cycle, and we compare its result with those obtained by the deterministic version of the model presented in Section 3.2.

4.1 Intuitions

In (Barrio *et al.*, 2006) a SSA-based *Delay Stochastic Simulation Algorithm* (DSSA) has been presented. In there, the authors present a novel stochastic algorithm for simulating biological systems where some of the reactions are associated with a delay. Since then, the algorithm has been applied to the simulation of mostly models of regulatory networks where delays are, for instance, used to model protein degradation (Bratsun *et al.*, 2005; Barrio *et al.*, 2006) or negative feedbacks (Cai, 2007).

In this section, we discuss the main ideas used as basis for the definition of that algorithm. First of all, we need to enhance the algebraic representation of reactions to reactions with delays. We recall that we assume a reaction with delay to be represented as $R \overset{k,\sigma}{\longmapsto} P$ where R, P and k have the usual meaning, the new parameter $\sigma \in \mathbb{R}$ represents the delay of the reaction. As we introduced in Section 2.4, to each reaction a stoichiometry vector is associated, hence we have for a reaction R_j a state-change vector ν_j. We augment the definition of the state-change vector as follows: let us denote each state-change vector ν_j as a the composition of the state-change vector for reactants, ν_j^r, and the state-change vector for products, ν_j^p, noting that the following equation holds

$$\nu_j = \nu_j^r + \nu_j^p \,.$$

For instance, let us assume a target system where three species A, B and C are involved; the state of the simulation is hence described by a 3-dimensional vector. We consider two reactions: the first consuming a molecule of species A and producing a molecule of species B and a molecule of species C, and the second consuming a molecule of species B and producing a molecule of the same species and one of species A. The reactions considered are

$$R_1 : A \overset{k_1,\sigma_1}{\longmapsto} B + C \qquad\qquad R_2 : B \overset{k_2,\sigma_2}{\longmapsto} B + A \,.$$

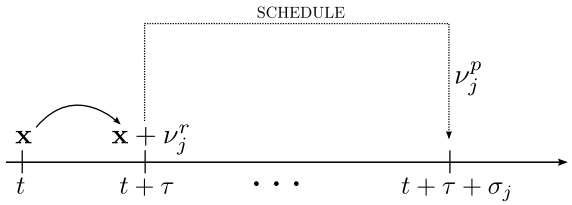

FIGURE 4.1: The semantics of the delay-as-duration approach.

Assuming the state vector to contain in the first entry the information on species A, in the second the one on B and in the third the one of C, we can define, for both R_1 and R_2 the state-change vectors for reactants and the one for products as follows

$$\nu_1^r = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \quad \nu_2^r = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \quad \nu_1^p = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad \nu_2^p = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

and, as expected, we have that

$$\nu_1 = \nu_1^r + \nu_1^p = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \qquad \nu_2 = \nu_2^r + \nu_2^p = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

As a consequence of this definition, the propensity function $a_j(\mathbf{x})$ introduced in Section 2.4.1 can be rephrased as follows

$$a_j(\mathbf{x}) = k \prod_{i=1}^{N} \binom{X_i(t)}{|\nu_{i,j}^r|}$$

where $\mathbf{X}(t) = \mathbf{x}$, $k \in \mathbb{R}$ denotes the kinetic function of reaction R_j and $|\nu_{i,j}^r|$ denotes the absolute value of the i-th coordinate of vector ν_j^r.

After introducing this simple extension of the notation for non-delayed systems, we can discuss the semantics of firing a reaction with delay. Let us assume the system to be in state $\mathbf{X}(t) = \mathbf{x}$, the system computes, accordingly to some technique, the putative time for the reaction to fire τ. We assume to fire a reaction R_j with delay σ_j. The system performs a move to state $\mathbf{X}(t+\tau)$ in which the reactants of the reaction have been removed from $\mathbf{x}$, namely

$$\mathbf{X}(t+\tau) = \mathbf{x} + \nu_j^r$$

and, in this context, the delays are used to *schedule* the insertion of the products at time $t+\tau+\sigma_j$. In Figure 4.1 a graphical representation of this semantics is given; in there the system is in state $\mathbf{x}$ at time t, jumps in state $\mathbf{x}+\nu_j^r$ at time $t+\tau$ and schedules the insertion of the products, by means of ν_j^p, at time $t+\tau+\sigma_j$ Obviously, if $\sigma_j \to 0$ this semantics becomes the classic semantics of firing a non-delayed reaction.

It is important to notice that the removal of the reactants and the insertion of the products are, in this case, two *detached events*. Practically, this implies that the reactants removed at time $t+\tau$ are excluded from any other possible interaction they could have by the firing of another reaction, in the whole time window $[t, t+\tau+\sigma_j]$. Notice how the interpretation of delays in DDEs differs from the one used in this algorithm. More precisely, in DDEs a delay states a dependancy of the derivative at time t with a past-state of the system whereas here a delay is such that a change in a future state of the system will depend on an event scheduled at present time. These two orthogonal views of delays will be related when discussing the mathematical foundations of this algorithm.

As the reactants are immediately removed from the system state, they cannot be involved in other reactions until the time at which the insertion of the products is reached, in this sense the

delay can be seen as a *duration* needed for the reactants to *exclusively* complete the reaction. Since this approach gives this interpretation of delays in the rest of the thesis we refer to it as the *delay-as-duration approach*. Such a naming of this interpretation of delays has been introduced in (Barbuti *et al.*, 2009b).

4.2 A DSSA with delay-as-duration approach (DDA)

In this section we introduce the formal definition of a DSSA *with delay-as-duration approach* (DDA) as firstly introduced in (Barrio *et al.*, 2006). In this definition we assume input systems where some of the reactions have a delay and others do not. The practical motivation for this hybrid algorithm is the fact that, in general, for most models some of the reactions involved are delayed and, quite often, these are not the majority of all. As a consequence, it is practical to think of a system with delays as a system in which some events are the classical modeled as, for instance, in the SSA.

Following (Barrio *et al.*, 2006), we classify reactions with delays into two categories: consuming and non-consuming reactions. The former class represents reactions where some of the reactants are consumed, while the latter class represents reactions where the reactants are also products. As an example, the two reactions

$$R_1 : A \overset{k_1,\sigma_1}{\longmapsto} B + C \qquad\qquad R_2 : B \overset{k_2,\sigma_2}{\longmapsto} B + A\,.$$

introduced in SECTION 4.1 can be classified as consuming, R_1, and non-consuming, R_2. Throughout the rest of this chapter, we denote the set of non-consuming reactions with delay by $\mathcal{R}_{nc}$, the set of consuming reactions with delay by $\mathcal{R}_c$, and the reactions without delays by $\mathcal{R}_{nd}$; notice that the whole set of the M reaction channels is $\mathcal{R} = \mathcal{R}_{nc} \cup \mathcal{R}_c \cup \mathcal{R}_{nd}$ and $\mathcal{R}_{nc}$, $\mathcal{R}_c$ and R_{nd} are pair-wise disjoint.

By adding delays to the SSA, the authors of (Barrio *et al.*, 2006) provide a method to model the firing of a reaction with delay based on the intuitions given in SECTION 4.1. We contextualize the general idea with respect to the classification of the reactions. First of all, given a system in state $\mathbf{X}(t) = \mathbf{x}$, the stochastic time quantity τ, meant to be the putative time for the next reaction to fire is computed exactly as in the SSA, namely as an exponentially distributed number such that

$$\tau \sim \mathcal{E}xp\,(a_0(\mathbf{x}))$$

where $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$ denotes the summation of all the evaluation of the propensity functions for both delayed and non-delayed reactions. Let us assume to choose to fire a generic reaction

$$(R_j : R \overset{k_j,\sigma_j}{\longmapsto} P) \in \mathcal{R}$$

where if R_j is non-delayed then $\sigma_j = 0$. We discuss the firing of the reactions dependently on their classification:

- (non-consuming, $R_j \in \mathcal{R}_{nc}$)

  This reaction has delay $\sigma_j > 0$ and since it is non-consuming, it happens that all the reactants hereby denoted as the multiset R are contained in the products denoted by the multiset P. As the reactants are not consumed, the reaction can be rewritten as

  $$R_j : \emptyset \overset{k_j,\sigma_j}{\longmapsto} (P \backslash R)$$

  where $\emptyset$ denotes the empty multiset of reactants (in this formulation no reactants are consumed hence algebraically $\emptyset$ corresponds to the null-vector) and $(P \backslash R)$ denotes the subtraction between multisets and, consequently, the effective number and type of molecules produced by this reaction. Notice that the two representation of the reaction are equivalent only under two conditions; firstly, in the evaluation of the propensity function $a_j(\mathbf{x})$, the contribution of the reactants in R is not lost and, secondly, the reactants are checked

for their presence in the system state. For instance, as the first condition is concerned, the non-consuming reaction R_2 presented in SECTION 4.1 can be represented as $R_2 : \emptyset \xmapsto{k_2,\sigma_2} A$ assuming its propensity function to be still defined as $a_2(\mathbf{x}) = k'[B]$. As regards the second condition, a reaction of the form $2A \xmapsto{k_3,\sigma_3} 2A + B$ which can be represented as $\emptyset \xmapsto{k_3,\sigma_3} B$ must have a propensity function evaluating to 0 in any state in which there is at most one A, which can be easily obtained defining the propensity by cases. Given this equivalent representation of non-consuming reactions, the semantics of firing the reaction requires to perform the following steps:

1. update clock to value $t + \tau$;
2. leave unmodified the state vector, hence $\mathbf{X}(t + \tau) = \mathbf{X}(t)$ since no reactants are consumed;
3. schedule the insertion of the products $(P \backslash R)$ (which can be represented as the vector $\nu_j^p + \nu_j^r$) at time $t + \tau + \sigma_j$.

We assume all the non-consuming reactions to be represented in this way, and hence in step 3 we schedule the insertion of the products described by ν_j^p, as defined in this alternative representation of the reaction.

- (consuming, $R_j \in \mathcal{R}_c$)

  This reaction has delay $\sigma_j > 0$ and since it is consuming its reactants have to be consumed by the firing of the reaction. In this case, by following the intuitions given in SECTION 4.1, the semantics of firing the reaction requires to perform the following steps:

  1. update clock to value $t + \tau$;
  2. update state vector, hence $\mathbf{X}(t + \tau) = \mathbf{X}(t) + \nu_j^r$ by consuming the reactants;
  3. schedule the insertion of the products by using ν_j^p at time $t + \tau + \sigma_j$.

- (non-delayed, $R_j \in \mathcal{R}_{nd}$)

  This reaction has delay $\sigma_j = 0$ and, as the DDA is based on the SSA, we use the classical semantics of the firing of non-delayed reactions in the SSA presented in Section 2.4.2, namely we perform the steps:

  1. update clock to value $t + \tau$;
  2. update state vector $\mathbf{X}(t + \tau) = \mathbf{X}(t) + \nu_j$ by removing the reactants and inserting the products;

  As expected, in this case the DDA performs no scheduling.

Before introducing the formal definition of this algorithm, we present some considerations regarding the scheduling of reactions. The notion of scheduling requires the definition of a policy for handling the scheduled events. In this algorithm such a policy is intuitively described as follows: if the DDA generates a putative time for next reaction τ such that there is at least a scheduled events in the time window $[t, t + \tau]$, then the reaction must be handled, and τ must be rejected. Of course, if this was not the case, then the algorithm would have forgotten all of those and this would have be controversial and incorrect.

We discuss now the formal definition of this policy in the DDA, let us consider the system to be at time t, and let us denote the set of all *scheduled events* as

$$S = \{(t', \nu') \mid t' \in \mathbb{R}, \nu' \in \mathbb{R}^n\}$$

where a pair (t', ν') denotes the fact that a reaction is scheduled to complete at time $t' > t$ and whose contribution is given by the vector ν', such a vector is the state-change vector of the products for a scheduled reaction. Among all the scheduled events in S, once that the algorithm

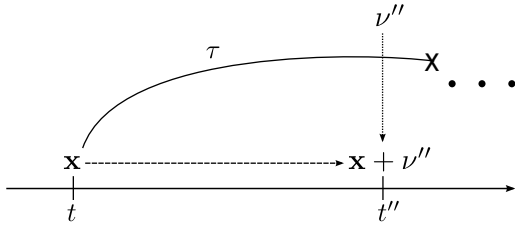

FIGURE 4.2: Handling scheduled reactions in the DDA.

has generated a value for τ, we want to concentrate on those pretending to complete in the time interval $[t, t+\tau]$. To this extent, let us define the set $S_{t,\tau} \subseteq S$ as

$$S_{t,\tau} = \{(t', \nu') \in S \mid t' \leq t + \tau\}$$

so that it represents all the events to be handled. Fairly intuitively, the DDA decides to handle the first event to complete, namely the one which has been scheduled before the others

$$(t'', \nu'') = \min\{S_{t,\tau}\} \implies \forall (t, \nu) \in S_{t,\tau}. t'' < t$$

where we assume the min operator evaluating on the left component of the pairs in $S_{t,\tau}$.

It is not guaranteed that such an event exists, indeed it does not exists only when there are no scheduled reaction in $[t, t+\tau]$ and hence $S_{t,\tau} = \emptyset$. However, if it exists the following operations have to be performed instead of the ones for firing a new reaction:

1. update clock to value t'';
2. update state vector $\mathbf{X}(t') = \mathbf{X}(t) + \nu''$ completing the firing of the scheduled reaction;
3. remove from the set S the entry (t'', ν'').

In FIGURE 4.2 the scenario is graphically represented. As a next step, the algorithm restart evaluating the propensity functions in this new state and re-generates a new value for τ. Notice that, as expected, $t'' > t$ and $t'' \leq t + \tau$ and notice that the putative time for the next reaction is discarded. This strategy which is completely new with respect to the SSA, must be proved to be correct.

The formal definition of the DDA is given in ALGORITHM 2. The DDA assumes as input an initial time t_0, a maximum simulation time T and an initial state $\mathbf{X}(t_0) = \mathbf{x}_0$. The DDA initialize the required structures in steps (1−3) and then, at each iteration, evaluates the propensity functions and generates τ and j exactly as the SSA does insteps $(5-8)$.

In steps $(9-13)$ a scheduled event is handled, in steps $(15-17)$ a non-delayed reaction is fired and, finally, in steps $(19-21)$ a delayed reaction is fired. Notice that, due to the representation we assumed for non-consuming reactions, we could define a unique step in the DDA to handle delayed reactions. Practically, if R_j is a non-consuming, its state-change vector for the reactants will be nil and, as a consequence, $\mathbf{x} = \mathbf{x} + \nu_k^r = \mathbf{x}$.

Since generating random numbers is a costly operation, other authors defined variants of the DSSA that avoid rejecting τ in the handling of scheduled reactions (Cai, 2007; Anderson, 2007). However, the interpretation of the delays used to define these variants is the same as that one we discussed here. In the next sections, we provide arguments to the correctness of this algorithm.

4.3 Mathematical foundations of the DDA

In the next two sections we discuss the mathematical foundations of the DDA. Firstly, we show the derivation of a *Delay Chemical Master Equation* (DCME) for target systems of this algorithm and, in the second section, we build the DDA from rigorous mathematical considerations leading to its correctness and exhibiting its connection with the master equation defined.

Algorithm 2 DSSA DDA(t_0, $\mathbf{x}_0$, T)

$t \leftarrow t_0$;
$\mathbf{x} \leftarrow \mathbf{x}_0$;
$S \leftarrow \emptyset$;
while $t < T$ **do**
  $a_0(\mathbf{x}) \leftarrow \sum_{j=1}^{M} a_j(\mathbf{x})$;
  let r_1, $r_2 \sim U[0,1]$;
  $\tau \leftarrow a_0(\mathbf{x})^{-1} \ln({r_1}^{-1})$;
  let j such that $\sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$;
  if $S_{t,\tau} \neq \emptyset$ **then**
    $(t', \nu') \leftarrow \min\{S_{t,\tau}\}$;
    $\mathbf{x} \leftarrow \mathbf{x} + \nu'$;
    $t \leftarrow t'$;
    $S \leftarrow S \backslash \{(t', \nu')\}$;
  else
    if $R_j \in \mathcal{R}_{nd}$ **then**
      $t \leftarrow t + \tau$;
      $\mathbf{x} \leftarrow \mathbf{x} + \nu_j$;
    else
      $\mathbf{x} \leftarrow \mathbf{x} + \nu_j^r$;
      $t \leftarrow t + \tau$;
      $S \leftarrow S \cup \{(t + \tau + \sigma_j, \nu_j^p)\}$;
    end if
  end if
end while

4.3.1 A Delay Chemical Master Equation for the DDA

In this section, we define a Delay Chemical Master Equation (DCME) for systems simulated by the DDA, obtained as an extension of the CME defined for the SSA. In (Barrio *et al.*, 2006) it is claimed that a DCME for such systems is defined, however accordingly to results we present in the next chapters it turns out that the equation defined in (Barrio *et al.*, 2006) models systems where to the firing of a reaction a unique state change is associated. For the sake of simplicity, we consider only systems where actions are delayed, since the extension of the result we present to the case of non-delayed reactions is straightforward. Although the interpretation of the delays as scheduling of reactions, this master equation contains delays expressed as dependancies on past-states of the system, as imposed by using the DDEs.

The same CME assumptions hold: we assume the system to be well-stirred, to be confined in a constant volume and to be in thermal equilibrium at some constant temperature. Let us assume a system initially described by $\mathbf{X}(t_0) = \mathbf{x}_0$ and let us assume an initial history function ω such that $\forall t < t_0$. $\mathbf{X}(t) = \omega(t)$, we consider a system where M reactions are present. Each reaction R_j is described by the two state-change vectors ν_j^r and ν_j^p, a delay σ_j and a propensity function $a_j(\mathbf{x})$; here $a_j(\mathbf{x})dt$ gives the probability of R_j to *start* in $\mathbf{x}$. Notice that, in the CME we say that such a quantity denotes the probability to fire the reaction but here, since start an completion are detached event, using the same terminology would be confusing.

Accordingly to the DDA, the state of the system changes every time a reaction starts (the reactants are removed), and every time a reaction completes (the products are inserted). We assume dt sufficiently small that the probability of starting and completing a reaction in the time $[t; t + dt)$ is negligible compared to the probability of at most one reaction starts or at most one completes. Moreover, reactions with delays are such that, when they start at time t, they complete at time $t + \sigma_j + dt$.

Let us denote the initial configuration of such a system as $I \equiv (\mathbf{x}_0, t_0; \omega)$, the probability $\mathbb{P}(\mathbf{x}, t + dt \mid I)$, is to be defined accordingly to the following cases:

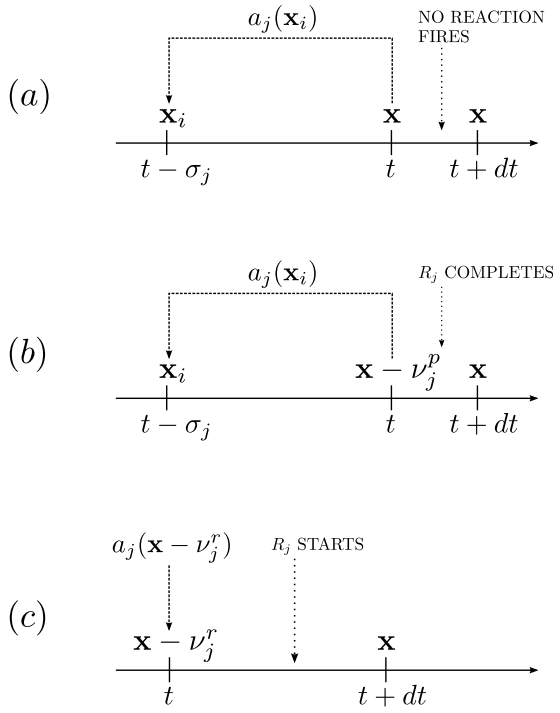

FIGURE 4.3: Events leading to the definition of the DCME for the DDA.

(a) at time t the system is in state $\mathbf{x}$ and no reaction either completes or start;

(b) at time t the system is in state $\mathbf{x}-\nu_j^p$ and reaction R_j completes;

(c) at time t the system is in state $\mathbf{x}-\nu_j^r$ and reaction R_j starts.

In FIGURE 4.3 the events leading to the definition of the DCME are graphically represented. We discuss now how to define the analytical form of the cases (a), (b) and (c). In order to define case (a), we recall that the probability that a reaction starts in state $\mathbf{x}$ is given by $a_j(\mathbf{x})dt$. Consequently, the probability that a reaction completes can be defined in terms of the probability of the reaction to have started in a past state of the system $\mathbf{x}_i$ (i.e. $a_j(\mathbf{x}_i)dt$) at time $t-\sigma_j$, if there exists a path between $\mathbf{x}_i$ and $\mathbf{x}$. Consequently, by denoting with $\mathbb{I}$ the set of all the states traversed by the system, case (a) is deÞned by the following analytical expression

$$\mathbb{P}(\mathbf{x},t\mid I)\left(1-\sum_{j=1}^{M}\sum_{\mathbf{x}_i\in\mathbb{I}}\mathbb{P}(\mathbf{x}_i,t-\sigma_j\mid\mathbf{x},t;I)a_j(\mathbf{x}_i)dt-\sum_{j=1}^{M}a_j(\mathbf{x})dt\right).$$

Notice that $\mathbb{P}(\mathbf{x}_i,t-\sigma_j\mid\mathbf{x},t;I)$ denotes the probability that the system is in state $\mathbf{x}_i$ at time $t-\sigma_j$ knowing that at time t it is in state $\mathbf{x}$, and knowing that the initial configuration is described by I. Notice that the need of coupling all the possible system states justifies the introduction of the set $\mathbb{I}$; in particular, for those states $\mathbf{x}_i$ which have no connection with state $\mathbf{x}$ such probability is 0.

Similarly, case (b) is defined by the following analytical expression

$$\sum_{j=1}^{M}\mathbb{P}(\mathbf{x}-\nu_j^p,t\mid I)\left(\sum_{\mathbf{x}_i\in\mathbb{I}}\mathbb{P}(\mathbf{x}_i,t-\sigma_j\mid\mathbf{x}-\nu_j^p,t;I)a_j(\mathbf{x}_i)dt\right)$$

and case (c) by

$$\sum_{j=1}^{M}\mathbb{P}(\mathbf{x}-\nu_j^r,t\mid I)a_j(\mathbf{x}-\nu_j^r)dt\,.$$

In the last equations, notice that the state is assumed to be $\mathbf{x} - \nu_j^r$ since in ν_j^r the components are negative. By using equation (2.6) on $\mathbb{P}(\mathbf{x}_i, t - \sigma_j \mid \mathbf{x}, t; I)$, we get the following equalities

$$\mathbb{P}(\mathbf{x}_i, t - \sigma_j \mid \mathbf{x}, t; I)\mathbb{P}(\mathbf{x}, t \mid I) = \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x}, t \mid I)$$
$$\mathbb{P}(\mathbf{x}_i, t - \sigma_j \mid \mathbf{x} - \nu_j^p, t; I)\mathbb{P}(\mathbf{x} - \nu_j^p, t \mid I) = \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x} - \nu_j^p, t \mid I)$$

and, by rearranging the equations, we get

$$\begin{aligned}\frac{\mathbb{P}(\mathbf{x}, t + dt \mid I) - \mathbb{P}(\mathbf{x}, t \mid I)}{dt} = &- \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x}, t \mid I) a_j(\mathbf{x}_i) \\ &- \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}, t \mid I) a_j(\mathbf{x}) \\ &+ \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x} - \nu_j^p, t \mid I) a_j(\mathbf{x}_i) \\ &+ \sum_{j=1}^{M} \mathbb{P}(\mathbf{x} - \nu_j^r, t \mid I) a_j(\mathbf{x} - \nu_j^r)\,.\end{aligned}$$

When considering the limit $dt \to 0$, we get the DCME

$$\begin{aligned}\frac{\partial \mathbb{P}(\mathbf{x}, t \mid I)}{\partial t} = &- \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x}, t \mid I) a_j(\mathbf{x}_i) \\ &- \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}, t \mid I) a_j(\mathbf{x}) \\ &+ \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x} - \nu_j^p, t \mid I) a_j(\mathbf{x}_i) \\ &+ \sum_{j=1}^{M} \mathbb{P}(\mathbf{x} - \nu_j^r, t \mid I) a_j(\mathbf{x} - \nu_j^r)\end{aligned} \tag{4.1}$$

which is a *Partial Delay Differential Equation* representing a set of DDEs describing the time-evolution of the probability of a system to occupy each one of a set of states, as it was for the CME. As it is similar to the CME, the same considerations are still valid; this equation is in general tractable only for simple systems and its solution is analytically hard to be computed. As in the case of the CME and the SSA, this motivated people in defining DSSAs.

4.3.2 Correctness of the DDA

In this section we want to show the correctness of the DDA, as we did for the SSA in SECTION 2.4.2. The key issue of the correctness of the DDA lies in proving the mathematical correctness of generating exponentially distributed numbers, handling scheduled reactions and choosing reactions to fire. In order to prove the correctness of these features, we try to use the results we recalled for SSA-based algorithms with time-dependent propensity functions. In particular, we try to define a proper version of equation (2.32) in the context of the DDA.

The intuition in using that equation comes from noting that the propensity functions of the DDA can be thought as time-dependent. In particular, as for the SSA, the key in generating simulated trajectories for $\mathbf{X}(t)$ when reaction have delay is neither the DCME nor even the function $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0; \omega)$, but rather a new function

$$p(\tau, j \mid \mathbf{x}, t; \boldsymbol{\Sigma})$$

defined such that $p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma})d\tau$ is the probability, given $\mathbf{X}(t) = \mathbf{x}$ and the scheduling list $\mathbf{\Sigma}$, that the next reaction in the system *starts* in the infinitesimal time interval $[t+\tau, t+\tau+d\tau)$ and is reaction R_j. Notice that $\mathbf{\Sigma}$ represents the reactions currently started but yet not completed, in this sense $\mathbf{\Sigma}$ is a time-dependent quantity depending on the particular realization of $\mathbf{X}(t)$ we consider, in fact in the DDA it is denoted as set S whereas here is $\mathbf{\Sigma}$. We know that, once a reaction starts, the system performs a state-change and the reaction is added to $\mathbf{\Sigma}$. It is fairly intuitive that, once a reaction from $\mathbf{\Sigma}$ completes, then the system performs a state-change, and then the propensity functions of the reaction, which are computed in a new state, change. Notice that differently from the quantity $p(\tau, j \mid \mathbf{x}, t)$ used in the SSA here we consider the fact that, since actions are non-instantaneous, such a probability depends *not only* on the current state $\mathbf{x}$, but also on the reactions currently running, namely those in $\mathbf{\Sigma}$. However, as in a non-delayed framework, such a function is the joint probability density function of two random variables, one modeling the putative time for the next reaction (τ), and the other modeling the choice of the next reaction to fire (j). In the following theorem we derive an analytical expression for such a function, proving the correctness of the DDA.

THEOREM 4.3.1 (DDA Correctness)**.** *The putative time for the next reaction is a continuous random variable τ whose density satisfies*

$$p(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) = \begin{cases} a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right) & \text{if} \quad 0 \leq \tau < \tau_M \\ p(\tau - \tau_M \mid \mathbf{x} + \nu', t', \mathbf{\Sigma}) & \tau \geq \tau_M \end{cases} \tag{4.2}$$

when $\mathbf{\Sigma} \equiv S$, $(t', \nu') = \min\{S_{t,\tau}\}$ and $\tau_M = (t' - t)$. Moreover, if $\tau < \tau_M$, the index of the next reaction to fire is a discrete random variable j with

$$P(j \mid \tau; \mathbf{x}, t; \mathbf{\Sigma}) = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}\,. \tag{4.3}$$

Proof. Similarly to equation (2.28), we start by denoting with $P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$ the probability that, given $\mathbf{\Sigma}$, no reactions start in the time interval $[t, t+\tau)$, so that we can write $p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma})$ as

$$p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma}) = P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) a_j(\tau) d\tau \tag{4.4}$$

where $a_j(\tau)$ denotes the fact that we evaluate the propensity function for reaction R_j in some state $\mathbf{x}'$ which is the state in which the system will be at future time $t+\tau$. Notice that such a state once that $\mathbf{\Sigma}$ and τ are known can be fully determined.

At this point, we are expected to have an analytical expression for $P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$. Indeed, we can define, similarly as we did for the CME for the SSA, an ODE for $P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$. Practically, we need to write the the probability $P_0(\tau + d\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$ in terms of $P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$ *and* the event "no reactions start in $[t+\tau, t+\tau+d\tau)$". The probability of a single reaction R_j to start at time $t+\tau$ is determined by the value $a_j(\tau)$, hence we are in the same situation of the SSA at least in deriving this equation and we can write

$$P_0(\tau + d\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) = P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})(1 - a_0(\tau)d\tau)\,.$$

When considering $d\tau \to 0$, we get the ODE

$$\frac{dP_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})}{d\tau} = -P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) a_0(\tau)\,. \tag{4.5}$$

This equation models the exponential decay of $P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$, as equation (2.29) was modeling such a decay for $P_0(\tau \mid \mathbf{x}, t)$, and its solution is

$$P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) = P_0(0 \mid \mathbf{x}, t; \mathbf{\Sigma}) \exp\left(-\int_0^\tau a_0(t')dt'\right)$$

In this case, as it was in the SSA, we have that $P_0(0 \mid \mathbf{x}, t; \mathbf{\Sigma}) = 1$ since it represents the probability that nothing happens in 0 time whatever $\mathbf{\Sigma}$ is. However, differently from the SSA the $-\int_0^\tau a_0(t')dt'$

can not be solved explicitly since the propensity functions of the reactions are not independent on the state-changes which could be induced by the completion of reactions in $\mathbf{\Sigma}$ and then we have that

$$P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) = \exp\left(-\int_0^{\tau} a_0(t')dt'\right).$$

Equation (4.4) can be rewritten by means of the analytical form of $P_0(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$

$$p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma}) = a_j(\tau)d\tau \exp\left(-\int_0^{\tau} a_0(t')dt'\right)$$

and then can be generalized to get the probability $p(\tau \mid \mathbf{x}, t; \mathbf{H})d\tau$ that a reaction fires at time $t+\tau$

$$p(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) = \sum_{j=1}^{M} p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma}) = a_0(\tau) \exp\left(-\int_0^{\tau} a_0(t')dt'\right). \tag{4.6}$$

This equation is what equation (2.32) corresponds to, in the context of the DDA. This implies that we need just to understand what equations (2.34) and (2.35) become in this context and then the theorem is proved.

In order to build both the equations, the biggest value φ such that $a_0(t')$ is constant for all the time window $[t, t+\varphi)$ is, in the this context, the time instant in which the first reaction to complete is in $\mathbf{\Sigma}$. To this extent, to have a precise value for φ, we assign a precise value to $\mathbf{\Sigma}$, so $\mathbf{\Sigma} \equiv S$, and S corresponds to the scheduling list of the DDA, for a particular realization of $\mathbf{X}(t)$. We then write $(t', \nu') = \min\{S_{t,\tau}\}$ so that the maximum step size for the functions is $\tau_M = (t' - t)$ as relative time, and t' as absolute time, so we have that $\varphi = \tau_M$. By writing a recursive definition for $a_0(t')$ where $\xi = a_0(\mathbf{x})$ we have that the two target equations become

$$\int_0^{\tau} a_0(\mathbf{x})dt' = \log({r_1}^{-1}) \tag{4.7}$$

if $\tau < \tau_M$, and

$$\int_0^{\tau-\tau_M} a_0(t')dt' = \log(cr_2^{-1}). \tag{4.8}$$

if $\tau > \tau_M$ and $\int_0^{\tau_M} a_0(\mathbf{x})dt' = \log(c^{-1})$. Notice that here we do not consider the case $\tau = \min\{S_{t,\tau}\}$ which in the DDA is embedded in equation (4.8). We can do that because of the continuity of the exponential distribution; indeed we do not lose probability information not considering such a case since by definition the point-wise probability of any continuous distribution is 0. These two equations determine equation (4.2), so conclude the first part of the proof.

The second part is much easier, in fact the generation of the index of the next reaction to fire with the same scheme used in the SSA comes from the fact that, if we write the conditional probability $P(j \mid \tau; \mathbf{x}, t; \mathbf{\Sigma})$ we have

$$P(j \mid \tau; \mathbf{x}, t; \mathbf{\Sigma}) = \frac{p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma})}{p(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})} = \frac{a_j(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right)}{a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right)} = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}$$

which is the same equation of the SSA rephrased in the context of the DDA, when $\tau < \tau_M$. □

By all these considerations it is clear that the choice of generating a value for τ as a sample of an exponential distribution with mean $a_0(\mathbf{x})$ is correct in step (7) of the DDA. Also, it is correct if $\tau \geq \tau_M$ to simply increase in steps $(11-12)$ the clock of the system to $t+\tau_M$, update the state and recompute a sample for an exponentially distributed number with mean given by computing the new values of the propensity functions. Another consideration is interesting, in equation (4.2) we have $\mathbf{x}+\nu'$ since, differently from equation (2.32), in this case we know the system we consider to follow the DDA semantics of firing actions.

An important final consideration is worth doing here. In proving the correctness, the crucial aspects of the DDA as the removal of the reactants at the start of a reaction did not play any special

role. In fact, we have been able to prove its correctness by defining the probabilistic event related to the start of a new reaction. On a practical side this means that the correctness of this algorithm is independent on the delay-as-duration approach; this intuition unravels the possibility of defining variants to this algorithm which can be proved to be correct with very similar arguments.

4.4 A DDA model of the cell cycle

In this section, we apply the DDA to the simulation of the tumour growth system we discussed in SECTION 3.2, and we compare the results obtained by using this algorithm with those obtained by using DDEs. In order to analyze the model we implemented the DDA in a tool named `DelaySim`.

Firstly, we need to define an equivalent stochastic model of the DDEs model for the tumour growth. We can do that by applying a delay correspondent to the one used to pass from the ODEs to the DDEs model, in order to get from the reaction-based model without delay a delayed one. If we consider the ODEs model firstly discussed in SECTION 3.2, we remark that the analogous reaction-based model is given by the reactions $T_{G_1} \overset{\alpha_1}{\longmapsto} T_S$, $T_S \overset{\alpha_2}{\longmapsto} T_{G_2}$, $T_{G_2} \overset{\alpha_3}{\longmapsto} T_M$ and $T_M \overset{\alpha_4}{\longmapsto} 2T_{G_1}$.

Of course as the ODEs one, this model is to be enriched by modeling other events as for instance cell death. The delay we used to create the DDEs model can be used to abstract the whole sequence of reactions

$$T_{G_1} \overset{\alpha_1}{\longmapsto} T_S \overset{\alpha_2}{\longmapsto} T_{G_2} \overset{\alpha_3}{\longmapsto} T_M$$

by using the a unique reaction describing the passage from the phase G_1 to phase M. By means of this consideration, we can define a stochastic model analogous to the DDEs one; the model describes the evolution of the two populations: cells in the interphase, T_I, and cells in the mitotic phase T_M. The events which can happen in the system are the following:

- a cell in the interphase becomes a cell in the mitotic phase with rate a_1, reaction (R_1);
- a cell in the mitotic phase splits with rate a_4 by producing two new cells in the interphase, reaction (R_2);
- a cell in the interphase dies with rate d_2, reaction (R_3);
- a cell in the mitotic phase dies with rate d_3, reaction (R_4).

The four reactions are formally represented as

$$(R_1)\ T_I \overset{a_1,\sigma}{\longmapsto} T_M \qquad (R_2)\ T_M \overset{a_4,0}{\longmapsto} 2T_I$$
$$(R_3)\ T_I \overset{d_2,0}{\longmapsto} \emptyset \qquad (R_4)\ T_M \overset{d_3,0}{\longmapsto} \emptyset$$

where the only reaction with delay is R_a. Accordingly to the DDA, the set of the reactions is built as follows

$$\mathcal{R}_c = \{R_1\} \qquad \mathcal{R}_{nd} = \{R_2, R_3, R_4\} \qquad \mathcal{R}_{nc} = \emptyset$$

where $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_{nd}$, and the propensity functions follow the law of mass action, hence have analytical definition

$$a_1 = a_1[T_I] \qquad a_2 = a_4[T_M]$$
$$a_3 = d_2[T_I] \qquad a_4 = d_3[T_M]\,.$$

Notice that the left a_1 denotes the name of the propensity function while the right a_1 denotes the kinetic constant of reaction R_1.

We considered the same parameters defined for the DDEs model and reported in TABLE 3.2 and we use the following input to the DDA

$$t_0 = 0 \qquad \mathbf{X}(t_0) = \begin{pmatrix} 10^5 \\ 10^5 \end{pmatrix}$$

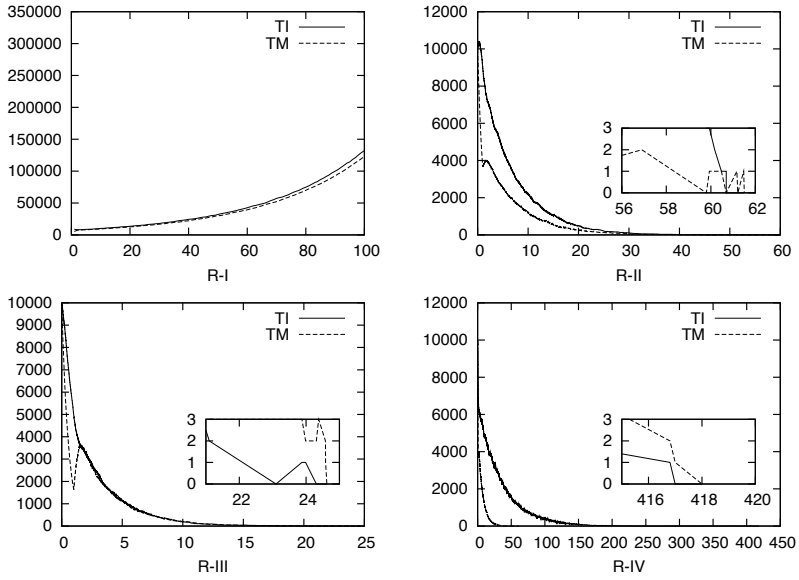

FIGURE 4.4: DDA simulations ($\sigma = 1$) for the regions of FIGURE 3.7.

We have run 100 simulations for each considered parameter setting. The results of simulations with the same parameters as those considered in FIGURES 3.8 and 3.9 are shown in FIGURES 4.4 and 4.5, respectively. In the figures we show the result of one randomly chosen simulation run for each parameter setting.

Qualitatively, results obtained with DDA simulations are the same as those obtained with numerical simulation of the DDEs, as shown in TABLE 3.1). Indeed, we have exponential tumor growth in region R-I, tumor decay in the other regions and oscillations arise when the delay is increased. However, from the quantitative point of view we have that in the DDA simulations the growth in region R-I and the decay in the other regions are always slower than in the corresponding numerical simulation of the DDEs. In fact, with $\sigma = 1$ by the numerical simulation of the DDEs we have that in region R-I after 100 days both the quantities of tumor cells in interphase and in mitotic phase are around 300000, while in the result of DDA simulations they are around 130000. In the same conditions, but with $\sigma = 10$, in the numerical simulation of the DDEs we have about 47000 tumor cells in mitosis and 57000 tumor cells in interphase, while in the DDA simulations we have about 5000 and 5500 cells, respectively.

In order to investigate this quantitative difference, we rephrased in this framework the same property defined for the DDEs, namely the first day of simulation in which the eradication of both the populations was visible:

$$\mathbf{X}(t^*) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

where

$$\forall t < t^*.\, \mathbf{X}(t) = \begin{pmatrix} n_I \\ n_M \end{pmatrix} \wedge n_I \geq 1 \ \wedge \ n_M \geq 1$$

and t^* is the ensemble of 100 stochastic runs of the DDA.

In TABLE 4.1 the average tumor eradication times obtained with DDA simulations are compared with those obtained with numerical simulation of the DDEs. Again, we have that in DDA simulations the dynamics is slower than in the numerical simulation of the DDEs. For instance, with $\sigma = 10$, in region R-IV the time needed for eradication in the DDEs is about 41% of the time needed in the DDA (440 against 1072), in region R-II the percentage is smaller, 26% (59 against 224), and, in region R-III, it reaches 9% (12 against 126). For the same regions with $\sigma = 1$ these differences are smaller but not negligible.

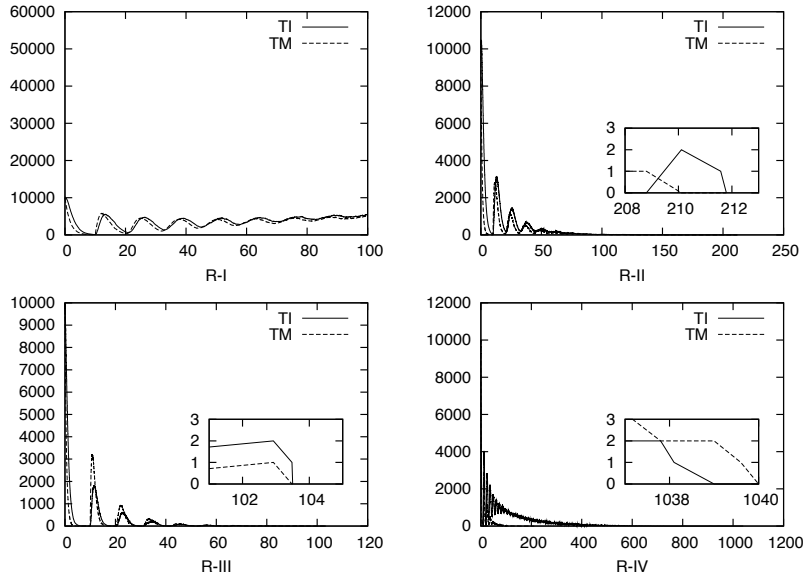

FIGURE 4.5: DDA simulations ($\sigma = 10$) for the regions of Figure 3.7.

	DDEs	DDA Simulation
R-II with $\sigma = 1.0$	50	64
R-II with $\sigma = 10.0$	59	224
R-III with $\sigma = 1.0$	15	29
R-III with $\sigma = 10.0$	12	126
R-IV with $\sigma = 1.0$	238	302
R-IV with $\sigma = 10.0$	440	1072

TABLE 4.1: Average eradication times given in *days* for DDEs model and DDA simulations.

The reason for such a difference is due to the definition of the semantics of the firing of reactions in the DDA. Such a semantics is such that the reactants are removed at the moment of scheduling a reaction to fire and, as a consequence, they can not have other interactions within the systems in the time interval elapsing between the scheduling and the handling of the scheduled event.

In this particular model this consideration becomes the following: a cell which started the passage from the interphase to the mitotic phase, can not be interrupted during such a event. Formally, the passage is regulated by the firing of reaction $R_1 : T_I \overset{a_1,\sigma}{\longmapsto} T_M$. The problem with this semantics is that, as we know from the model specification, a cell may die, in any phase of the cell cycle. This appears in the model as the two reactions R_3 and R_4. As a consequence, to a cell consumed by reaction R_1 reaction R_3 can not be applied anymore and reaction R_4 can be applied only when the cell has fully performed the phase passage and is its mitosis.

This type of problem does not exist, in the DDEs where the delays are dependancies on past states of the system. By this consideration, it is clear that a property concerning time as the one we tried to observe on the traces of the system, is observed with a delay with respect to the time in which it really happens. Furthermore, even if in this particular example this problem induces only more delay in observing the property, it is not to be excluded that, in a more complex model, such a semantics could affect the whole dynamics of the system leading to possibly wrong behaviors of the system.

In general, we claim that the DDA approach is suitable only for those types of systems in which it is plausible to have reactants that do not have any interaction during the firing of delayed

reactions. In fact, for systems in which this assumption does not hold, it is necessary to have a different approach to the firing of such reactions and, in the next chapters, we present some alternatives. The tumor growth system we discussed here is an example of systems where this assumption does not hold. Differently, the cellular models discussed in SECTION 3.1.1 are likely to be analyzed by the DDA since, in those models, accordingly to (Bratsun *et al.*, 2005; Barrio *et al.*, 2006; Cai, 2007) it is biologically plausible that both DNA and mRNA during transcription and translation, can not perform other simultaneous reactions.

Chapter 5

A purely delayed approach

In this chapter we introduce a new DSSA for the simulation of biological systems with delays. The idea behind this algorithm is the opposite of the one at the basis of the DDA: reactants involved in a reaction with delay can have other interactions before the completion of the delayed reaction. Of course, the fact that the reactants are not locked requires to define proper policies to have a consistent simulation; we discuss all of these aspects here and in CHAPTER 7.

Firstly, we discuss the motivations for presenting this new approach and then we give a naïve definition of the new algorithm. Although this first definition is naïve, it is expressive enough to justify this new DSSA, as we prove by analyzing the cell cycle model with this new algorithm and comparing the result with DDEs and DDA simulation. We discuss its mathematical foundation by defining a master equation for systems obeying this interpretation of the delays, and we prove the correctness of the algorithm.

5.1 Intuitions

In the previous chapter we discussed a DSSA based on the delay-as-duration approach, we compared the DDEs cell-cycle model with the stochastic model simulated by the DDA. In the comparison between the results obtained by using DDEs and the DDA different results have been observed. We argued that the motivation for observing such quantitative different behaviors was the semantics of the firing delayed reactions in the DDA. The problem with that semantics was the fact that the reactants don not have any interaction during the firing of delayed reactions. In fact, for systems in which this assumption does not hold, it is necessary to have a different approach to the firing of such reactions and, in this section, we present a new semantics for firing delayed reactions.

The novel approach we propose consists in firing a reaction completely when its associated scheduled events is handled, namely removing its reactants and inserting its products after the delay. Formally, this means that for a generic reaction

$$R_j : R \xmapsto{k,\sigma} P$$

represented in the same notation used for the DDA, by assuming the system to be $\mathbf{X}(t) = \mathbf{x}$, the putative time for the next reaction to be τ, the system state at time $t + \tau$ is the same at time t, namely holds

$$\mathbf{X}(t + \tau) = \mathbf{X}(t) .$$

Also in this context the delays are used to *schedule* the removal of the reactants and the insertion of the products at time $t + \tau + \sigma_j$. Obviously, the notion of scheduling here is the same used in the DDA however the operations performed to handle scheduled events will be definitely different. The fact that we simply schedule delayed reactions without immediately removing their reactants motivates the terminology of *purely delayed* (Barbuti *et al.*, 2009b).

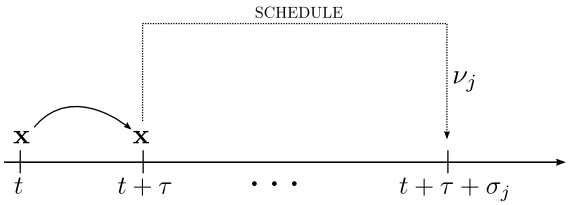

FIGURE 5.1: The semantics of the purely delayed approach.

It is important to notice that the removal of the reactants and the insertion of the products are, in this case, one *unique event* as in the SSA. Practically this new semantics, even if it is a fairly intuitive variant of the delay-as-duration, is such that the reactants removed at time $t + \tau$ can participate in other possible interaction they could have by the firing of another reaction, in the whole time window $[t, t + \tau + \sigma_j]$. Of course, this new capability for the reactants will require a more careful handling of scheduled events, as we shall see in the formal definitions of the DSSAs we are going to present.

There are some similarities between the delay-as-duration and this semantics in fact, also in this case, we have that the delays are treated as scheduling (instead that as past-state dependancy as in the DDEs) and, furthermore, when delays are 0 this semantics reduces exactly to the one of the SSA. In FIGURE 5.1 a graphical representation of this semantics is given.

5.2 A DSSA with a purely delayed approach (PDA)

In this section we give a formal definition of a naïve DSSA *with a purely delayed approach* (PDA), and in the next sections its mathematical foundations are discussed. The reason we claim that this is a naïve version are investigated in SECTION 7.1 of CHAPTER 7.

In the PDA we try to overcome the fact that in the DDA the reactants cannot have other interactions. This interpretation of delays was firstly implicitly adopted in (Bratsun *et al.*, 2005), to model a very simple example of protein degradation. As in the DDA, we assume constant delays defined by a real number $\sigma \geq 0$. Indeed, also the PDA deals with systems where some of the reactions have a non-zero delay and other are non-delayed. However, differently from the DDA, we use the same interpretation of delays to define the method for firing both non-consuming and consuming reactions. We classify reactions into two categories, delayed and non-delayed, denoted as $\mathcal{R}_d$ and $\mathcal{R}_{nd}$, respectively. The set of all the M reactions is denoted as $\mathcal{R} = \mathcal{R}_d \cup \mathcal{R}_{nd}$ and $\mathcal{R}_d$ and $\mathcal{R}_{nd}$ are pair-wise disjoint.

We discuss now the formal operations performed by the PDA, as we did for the DDA. First of all, given a system in state $\mathbf{X}(t) = \mathbf{x}$, the putative time for the next reaction to fire τ is computed exactly as in the SSA (and as in the DDA)

$$\tau \sim \mathcal{E}xp\left(a_0(\mathbf{x})\right)$$

where $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$ denotes the summation of all the evaluation of the propensity functions for both delayed and non-delayed reactions. Let us assume to choose to fire a generic reaction

$$(R_j : R \xmapsto{k_j,\sigma_j} P) \in \mathcal{R}\,.$$

We discuss the firing of the reactions dependently on their classification:

- (delayed, $R_j \in \mathcal{R}_d$)

  This reaction has delay $\sigma_j > 0$ and the semantics of firing the reaction requires to perform the following steps:

  1. update clock to value $t + \tau$;

2. leave unmodified the state vector, hence $\mathbf{X}(t+\tau) = \mathbf{X}(t)$ since this is purely delayed;
3. schedule the removal of the reactants and the insertion of the products by using the state-change vector ν_j at time $t + \tau + \sigma_j$.

- (non-delayed, $R_j \in \mathcal{R}_{nd}$)

  This reaction has delay $\sigma_j = 0$ and, as in the DDA and in the SSA, we use the classical semantics of firing reactions in the SSA:

  1. update clock to value $t + \tau$;
  2. update state vector $\mathbf{X}(t+\tau) = \mathbf{X}(t) + \nu_j$ by removing the reactants and inserting the products;

  As expected, in this case the PDA performs no scheduling, as the DDA.

Notice that non-consuming reactions, although non classified in the PDA, are handled in the same way by the DDA and the PDA when assuming the alternative representation discussed in the definition of the DDA.

The PDA implied a notion of scheduling equivalent to the one used in the DDA and hence we do not discuss it here. However, in the PDA it may happen that, when handling a scheduled reaction, the reactants may not be present in the current state. In fact, they could have been transformed (or drastically destroyed) by other interactions happened after the scheduling. In this case, the scheduled reaction has to be *ignored* as, in some sense, it is like it has been interrupted.

To formalize this, we know that a reaction can be applied only if its reactants are all present in the current state of the simulation. If it was not so, the application would lead to negative populations, a problem arising in the approximated SSAs when multiple firings are collapsed in a unique firing or in deterministic models, either delayed or non-delayed, as shown in FIGURE 3.9. Algebraically, applicability corresponds to the fact that, given $\mathbf{X}(t) = \mathbf{x}$ and a reaction to fire R_j,

$$\nu_j^r \prec \mathbf{x}$$

where ν_j^r is the state-change vector of the reactants of reaction R_j and $\prec$ is the ordering relation defined as

$$\mathbf{X}(t) = \begin{pmatrix} X_1(t) \\ \dots \\ X_N(t) \end{pmatrix} \implies \forall i = 1, \dots, N. \; -\nu_{ij}^r \leq X_i(t)\,.$$

since ν_{ij}^r is the number of reactants of species $X_i(t)$ consumed by R_j. In order to verify that a scheduled reaction can effectively fire, in this naïve definition of the PDA we simply check whether this condition holds. In order to record the information of the vector of the reactants to check we extend the scheduling set S presented in the DDA to triples of the form

$$S = \{(t, \nu, \nu^r) \mid t \in \mathbb{R}, \nu \in \mathbb{R}^n, \nu^r \in \mathbb{R}^n\}$$

where t and ν have the usual meaning and ν^r denotes the reactants state-change vector. The set $S_{t,\tau}$ is not affected in its definition and construction by this new definition of S.

The formal definition of the PDA is given in ALGORITHM 3. The PDA assumes the same DDA input: an initial time t_0, a maximum simulation time T and an initial state $\mathbf{X}(t_0) = \mathbf{x}_0$. Steps $(1-8)$ are exactly the same of the DDA. Steps $(9-15)$ are used to handle a scheduled event where steps $(11-13)$ handle an applicable reaction. Independently on the applicability of the reaction the event is removed from S in step (15). Steps $(17-19)$ fire a non-delayed reaction, as it was for the analogous steps in the DDA and, finally, steps $(21-22)$ schedule the firing of a delayed reaction.

Algorithm 3 DSSA PDA(t_0, $\mathbf{x}_0$, T)

$t \leftarrow t_0$;
$\mathbf{x} \leftarrow \mathbf{x}_0$;
$S \leftarrow \emptyset$;
while $t < T$ **do**
  $a_0(\mathbf{x}) \leftarrow \sum_{j=1}^{M} a_j(\mathbf{x})$;
  let r_1, $r_2 \sim U[0,1]$;
  $\tau \leftarrow a_0(\mathbf{x})^{-1} \ln({r_1}^{-1})$;
  let j such that $\sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$;
  if $S_{t,\tau} \neq \emptyset$ **then**
    $(t', \nu', \nu'^r) \leftarrow \min\{S_{t,\tau}\}$;
    if $\nu'^r \prec \mathbf{x}$ **then**
      $\mathbf{x} \leftarrow \mathbf{x} + \nu'$;
    end if
    $t \leftarrow t'$;
    $S \leftarrow S \backslash \{(t', \nu', \nu'^r)\}$;
  else
    if $R_j \in \mathcal{R}_{nd}$ **then**
      $t \leftarrow t + \tau$;
      $\mathbf{x} \leftarrow \mathbf{x} + \nu_j$;
    else
      $t \leftarrow t + \tau$;
      $S \leftarrow S \cup \{(t + \tau + \sigma_j, \nu_j, \nu_j^r)\}$;
    end if
  end if
end while

5.3 Mathematical foundations of the PDA

In the next two sections we discuss the mathematical foundations of the PDA, as we did for the DDA in SECTION 4.3. Firstly, we show the derivation of a Delay Chemical Master Equation (DCME) for target systems of this algorithm and, in the second section, we build the PDA from rigorous mathematical considerations leading to its correctness and exhibiting its connection with the master equation defined.

5.3.1 A Delay Chemical Master Equation for the PDA

In this section, we define a DCME for systems simulated by the PDA, again obtained as an extension of the CME defined for the SSA. In the same style of equation (4.1) and the DDA, this master equation contains delays expressed in the same fashion of DDEs, namely as dependancies on past-states of the system.

We assume the same scenario assumed for the DCME of the DDA. However, differently from the DDA, in the PDA the state of the system changes only when a reaction completes (reactants are removed and products are inserted). As in the DCME for the DDA, let us assume dt sufficiently small that the probability of starting and completing a reaction in the time $[t; t + dt)$ is negligible compared to the probability to start or complete at most one reaction. As in the DDA reactions with delays are such that, when they start at time t, they complete at time $t + \sigma_j + dt$.

By denoting with I the initial configuration $\mathbf{x}_0, t_0, \omega$ the probability $\mathbb{P}(\mathbf{x}, t + dt \mid I)$ is defined accordingly to the following cases:

(a) at time t the system is in state $\mathbf{x}$ and no reaction neither completes nor start;

(b) at time t the system is in state $\mathbf{x} - \nu_j$ and reaction R_j completes;

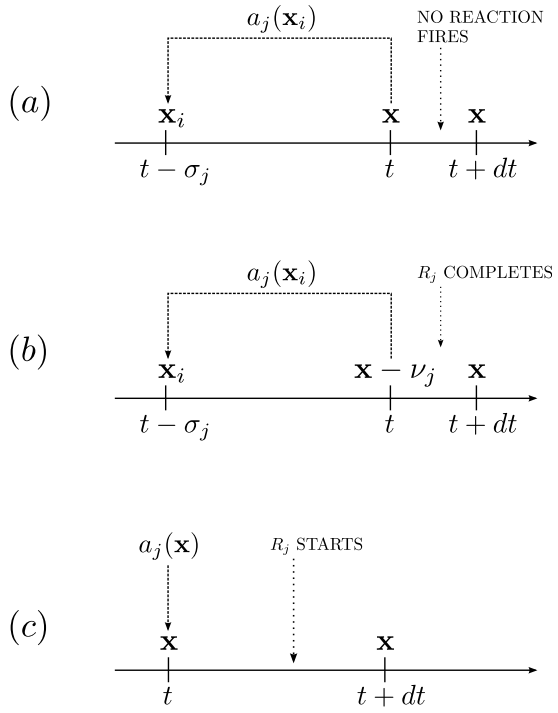

FIGURE 5.2: Events leading to the definition of the DCME for the PDA.

(c) at time t the system is in state $\mathbf{x}$ and reaction R_j starts.

In FIGURE 5.2 the events leading to the definition of the DCME are graphically represented. We discuss now how the define the analytical form of the cases (a), (b) and (c). Case (a) is equivalent to case (a) defined in the DCME for the DDA, hence we do not discuss it here. Case (b) is defined by the following analytical expression

$$\sum_{j=1}^{M} \mathbb{P}(\mathbf{x}-\nu_j, t \mid I) \left(\sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j \mid \mathbf{x}-\nu_j, t; I) a_j(\mathbf{x}_i) dt \right)$$

where the difference from the DCME for the DDA is that here we assume the system to be in state $\mathbf{x}-\nu_j$ instead of $\mathbf{x}-\nu_j^p$ since here the reactions fire accordingly to the PDA.

Case (c) is defined by

$$\mathbb{P}(\mathbf{x}, t \mid I) \left(\sum_{j=1}^{M} a_j(\mathbf{x}) dt \right)$$

and it differs from case (c) in the DCME for the DDA because we assume to be in state $\mathbf{x}$ instead of state $\mathbf{x}-\nu_j^r$, again as a consequence of the PDA interpretation.

In order to derive the DCME we use the same theorem used in the derivation of the DCME of

the DDA, and hence we can write

$$\begin{aligned}\frac{\mathbb{P}(\mathbf{x}, t+dt \mid I) - \mathbb{P}(\mathbf{x}, t \mid I)}{dt} = &- \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j; \mathbf{x}, t \mid I) a_j(\mathbf{x}_i) \\ &- \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}, t \mid I) a_j(\mathbf{x}) \\ &+ \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j; \mathbf{x}-\nu_j, t \mid I) a_j(\mathbf{x}_i) \\ &+ \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}, t \mid I) a_j(\mathbf{x}) \,.\end{aligned}$$

and, by simplifying $\sum_{j=1}^{M} \mathbb{P}(\mathbf{x}, t \mid I) a_j(\mathbf{x})$ and considering the limit $dt \to 0$, we get the DCME

$$\begin{aligned}\frac{\partial \mathbb{P}(\mathbf{x}, t \mid I)}{\partial t} = &- \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j; \mathbf{x}, t \mid I) a_j(\mathbf{x}_i) \\ &+ \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j; \mathbf{x}-\nu_j^p, t \mid I) a_j(\mathbf{x}_i) \,. \end{aligned} \tag{5.1}$$

As expected, this equation is a *Partial Delay Differential Equation* representing a set of DDEs describing the time-evolution of the probability of a system to occupy each one of a set of states, as it was for the other master equations. Also, the same considerations about its intractability are still valid, as they hold for the DCME for the DDA. Finally, the fact that the DDA and the PDA denote stochastic processes whose densities are the solution of two different master equations gives a stronger mathematical result to compare the two algorithms and supports the results outlined in the comparison performed in the SECTION 5.4.

5.3.2 Correctness of the PDA

In this section we want to investigate the correctness of this algorithm as we did for the DDA in SECTION 4.3.2.

Let us consider the same probability functions $p(\tau \mid \mathbf{x}, t; \mathbf{\Sigma})$ introduced to mathematically justify the DDA. We can state this theorem for the PDA.

THEOREM 5.3.1 (PDA Correctness)**.** *The putative time for the next reaction is a continuous random variable τ whose density satisfies*

$$p(\tau \mid \mathbf{x}, t; \mathbf{\Sigma}) = \begin{cases} a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right) & \text{if} \quad 0 \le \tau < \tau_M \\ p(\tau - \tau_M \mid \mathbf{x} + \nu', t', \mathbf{\Sigma}) & \text{if} \quad \tau \ge \tau_M \wedge \nu'^r \prec \mathbf{x} \\ p(\tau - \tau_M \mid \mathbf{x}, t', \mathbf{\Sigma}) & \text{if} \quad \tau \ge \tau_M \wedge \nu'^r \not\prec \mathbf{x} \,. \end{cases} \tag{5.2}$$

when $\mathbf{\Sigma} \equiv S$, $(t', \nu', \nu'^r) = \min\{S_{t,\tau}\}$ and $\tau_M = (t' - t)$. Moreover, if $\tau < \tau_M$, the index of the next reaction to fire is a discrete random variable j with

$$P(j \mid \tau; \mathbf{x}, t; \mathbf{\Sigma}) = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})} \,. \tag{5.3}$$

Proof. The result comes from simple considerations: the crucial quantity, $p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma})$ denotes the probability of the next reaction to start at time τ, given current state $\mathbf{X}(t) = \mathbf{x}$ and a set of reactions currently running $\mathbf{\Sigma}$. As we stated after deriving equations (4.7), (4.8) and (4.3) the

crucial aspects of the DDA (i.e. removal of the reactants at the start of a reaction) did not affect the algebraic form of the probabilities involved; in fact, we claimed that the proof was a more general result. In particular, we proved the correctness of a scheduling algorithm working with the schema shared by the DDA and the PDA.

As a consequence of this, equations (4.7), (4.8) and (4.3) are still valid in the context of the PDA and provide its correctness in the very same way. Only one consideration is worth in the case of the PDA; the algorithm performs steps in which $S_{t,\tau} \neq \emptyset$ which implies $\tau > t'$ if $(t', \nu', \nu'^{r}) = \min\{S_{t,\tau}\}$ and then checks if $\nu'^{r} \prec \mathbf{x}$. Only if such check is true the reaction is applied but, in any case, the clock is increased to t'. At a first sight, it may seem that this sequence of events is not part of the probability defined by equations (4.7) and (4.8) but this is not correct. In fact, in the derivation of the two equations we assume the propensity functions to be recursively defined as $a_0(t'') = a_0(\mathbf{x})$ if $t'' < t'$ and $a_0(t'')$ otherwise, and there are no constraints to impose that $a_0(t'') \neq a_0(\mathbf{x})$ which is our scenario when we reject to apply a scheduled reaction; as a consequence our choices are modeled by equation (4.8) even in this case. □

5.4 A PDA model of the cell cycle

In this section we prove the utility of the PDA by comparing the DDEs, the DDA and the PDA on the tumour growth model analyzed in SECTION 4.4 by using the DDA. Even if the definition of the PDA is naïve, as we shall see in the next, it contains enough of the ideas behind the purely delayed approach. Another reason for using this naïve definition of the PDA is that it permits an easy implementation; as for the DDA, we implemented this PDA in the JAVA tool `DelaySim`.

In order to analyze the tumour growth model we notice that the input similarity between the DDA and the PDA is such that we can use the same set of reactions to perform PDA simulations

$$(R_1)\ T_I \xmapsto{a_1,\sigma} T_M \qquad (R_2)\ T_M \xmapsto{a_4,0} 2T_I$$
$$(R_3)\ T_I \xmapsto{d_2,0} \emptyset \qquad (R_4)\ T_M \xmapsto{d_3,0} \emptyset$$

which are here classified as

$$\mathcal{R}_d = \{R_1\} \qquad \mathcal{R}_{nd} = \{R_2, R_3, R_4\}.$$

We considered the same parameters defined for both DDEs and DDA and reported in TABLE 3.2 and we used the same input of the DDA

$$t_0 = 0 \qquad \mathbf{X}(t_0) = \begin{pmatrix} 10^5 \\ 10^5 \end{pmatrix}.$$

As for the DDA, we have run 100 simulations of the stochastic model of tumor growth for each considered parameter setting. The results of simulations (we refer to these simulations as PDA simulations) with the same parameters as those considered in FIGURE 3.8 and in FIGURE 3.9 are shown in FIGURE 5.3 and in FIGURE 5.4, respectively. In the figures we show the result of one randomly chosen simulation run for each parameter setting.

Qualitatively, results obtained with PDA simulations are the same as those obtained with numerical simulation of the DDEs (and with DDA simulations). From the quantitative point of view we have that in the PDA simulations the growth in region R-I with $\sigma = 1$ is almost equal to the corresponding numerical simulation of the DDEs (about 300000 tumor cells in both mitosis and interphase after 100 days, we recall that the DDA had reached values around 130000). On the contrary, with $\sigma = 10$, the difference between DDEs and PDA is higher: we have about 22000 tumor cells in interphase against 57000 for the DDEs and 5500 for the DDA, and 16000 tumor cells in mitosis against 47000 for the DDEs and 5000 for the DDA.

In order to investigate this quantitative difference, we recomputed in this framework the same property defined for the DDA, namely the first day of simulation in which the eradication of both

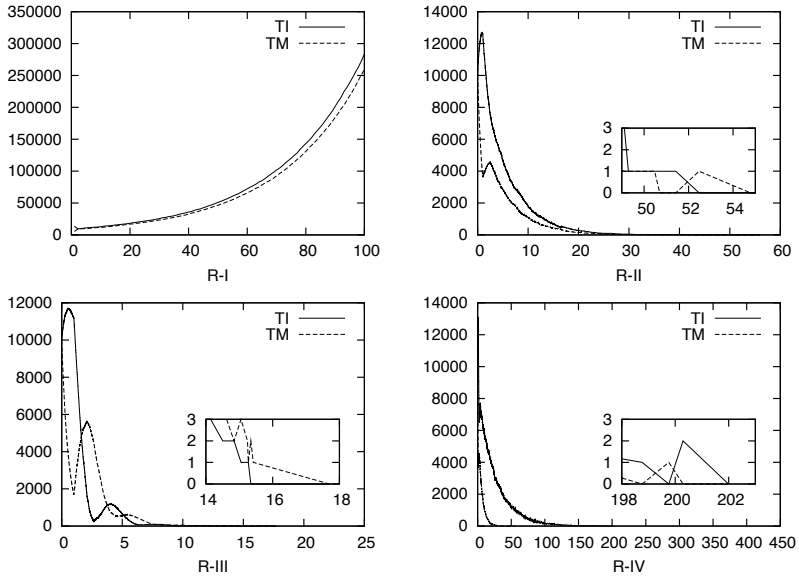

FIGURE 5.3: PDA simulations ($\sigma = 1$) for the regions of FIGURE 3.7.

	DDEs	DDA Simulation	PDA Simulation
R-II with $\sigma = 1.0$	50	64	51
R-II with $\sigma = 10.0$	59	224	67
R-III with $\sigma = 1.0$	15	29	17
R-III with $\sigma = 10.0$	12	126	20
R-IV with $\sigma = 1.0$	238	302	214
R-IV with $\sigma = 10.0$	440	1072	248

TABLE 5.1: Average eradication times given in *days* for DDE model, DDA and PDA stochastic models. For the stochastic models the entries represent the sample of 100 simulations.

the populations was visible:

$$\mathbf{X}(t^*) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

where

$$\forall t < t^*.\, \mathbf{X}(t) = \begin{pmatrix} n_I \\ n_M \end{pmatrix} \wedge n_I \geq 1 \ \wedge \ n_M \geq 1$$

and in this case t^* is the ensemble of 100 stochastic runs of the PDA. In TABLE 5.1 the average tumor eradication times obtained with PDA simulations are compared with those obtained with numerical simulation of the DDEs and the DDA. In PDA simulations the dynamics is generally slower than in the numerical simulation of the DDEs but it is faster than the DDA one. With $\sigma = 10$, in region R-IV the time needed for eradication in the PDA is smaller than the one in the DDEs (248 days against 440, DDA is 1072). In region R-II the values are: 67 days for the PDA and 59 days for the DDEs, DDA is 224. In region R-III values are: 20 days for the PDA, 12 days for the DDEs, and 126 days for DDA.

By analyzing this data, it is clear that this approach is more suitable to the simulation of systems in which the reactants can have multiple interactions during the firing of reactions with delay.

It is important to remark that differences between delay stochastic simulation results and

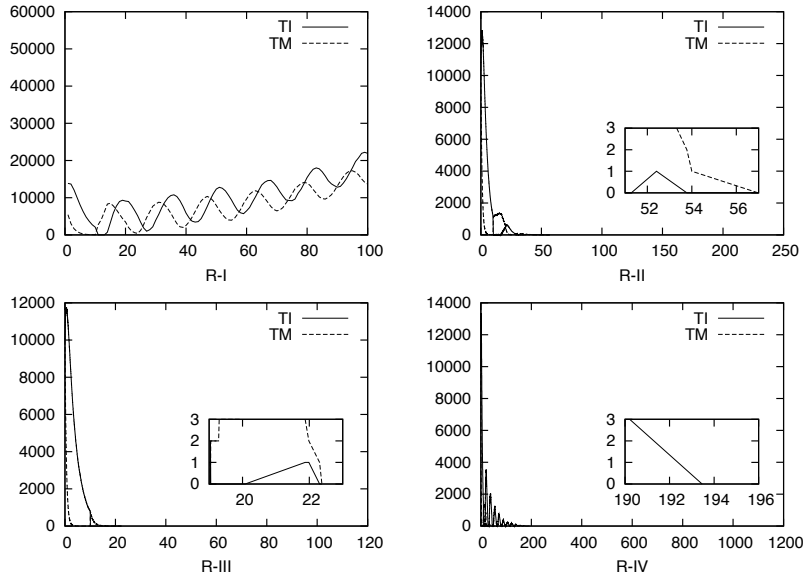

FIGURE 5.4: PDA simulations ($\sigma = 10$) for the regions of FIGURE 3.7.

numerical solutions of DDEs are also influenced by the initial conditions. The numerical solution of the DDEs assumes the initial population to be constant and greater than zero in the time interval $[-\sigma, 0]$. This allows delayed event to be enabled in the time interval $[0, \sigma]$. Both the DDA and the PDA start to schedule delayed events from time 0, hence delayed reactions can fire only after the time σ. This results, when σ is great enough, in a behavior that is, in general, delayed with respect to that given by the DDEs.

Summarizing, this algorithm seems more appropriate for simulating systems in which it is correct that reactants involved in a delayed reaction can have multiple simultaneous interactions. In this sense, epidemic models introduced in SECTION 3.1.2 are amenable to be simulated by the PDA since, in general, it is reasonable to think of susceptible to be able have other interactions when the infective process is in progress. Differently, evolutionary models of SECTION 3.1.3 would require a more complex interpretation of the delays. Such an interpretation could by created by allowing both the interpretation in a unique reaction. In fact, in there a predated prey should be immediately removed from the state since, as we know, it is killed by the predator. Moreover, the predator should be able to perform other simultaneous reactions before introducing a new juvenile.

Chapter 6

A history-dependent algorithm

In this chapter we propose a novel DSSA which is based on a completely different interpretation of the delays. Indeed, this algorithm is inspired in its definition by the use of delays in the DDEs, namely by the fact that delays are used to define dependancies on the past-states of the system rather than on the scheduling, with respect to some policy, of reactions. The flip-coin for this algorithm is the space required with respect to the previously presented DSSAs, in fact the space required to store history data is more than the one used to store scheduling data.

As for all the other DSSAs we prove the correctness of the new algorithm and we define a Delay Chemical Master Equation for target systems this algorithm and we show that it is the same defined in SECTION 5.3.1 for the PDA, leading to the fact that the two algorithms are equivalent since the densities of the underlying stochastic processes are the solution of the same master equation.

6.1 Intuitions

In the previous sections we presented some DSSAs, which are all based on the idea of modeling delays accordingly to some scheduling policy. In this section, we try to investigate whether a framework such as DDEs, which is based on a completely different notion of delays, can be of help in inspiring a DSSA where the delays are represented as dependencies on past-states of the system.

This task is not prohibitive, in particular, if we want to extend to SSA differently from the previously presented DSSAs, then the only portion of the algorithm which depends on state of the system and is an invariant in the SSA and all other DSSAs, is the way in which the propensity functions are evaluated. In particular, the question we have to address is whether we can evaluate such functions in past states of the system, and use those values to determine choices to be performed in the current system state. As expected, this corresponds logically to what DDEs are based on. In fact, in DDEs the derivatives of the unknown function at the current time depends on the function evaluated in past-states of the system.

The natural way to define this new DSSA, also considering the definition of the DCMEs for the previously presented DSSAs, is based on the following idea: once we are in a state $\mathbf{x}$ at time t, we evaluate each propensity function in a past state $\mathbf{x}'$ if the system was in state $\mathbf{x}'$ at time $t - \sigma_j$. Subsequently, we can generate the putative time for the next reaction to fire as the sampling of a proper random variable. What such a distribution of the variable is, and the possible consequences of sampling a specific value from such a variable is topic for the next sections. Intuitively, as in the DSSAs based on scheduling, the values sampled at each step of the algorithm determined, by using some conditions, possibly different behaviors. In this algorithm, we expect to have similar conditions which determine similar behaviors.

Finally, once a putative time for the next reaction has been chosen, and once that the next reaction to fire is chosen, then we can simply fire the reaction by using the classical semantics of the SSA. Of course, the fact that we compute probabilities of firing reaction in the current state by means of probabilities evaluated in past-states requires, in general, to define firing conditions to

avoid inconsistencies such as negative populations. Notice that this makes immediately noticeable that, if this new algorithm must be equivalent to some other previously presented DSSA, then it is more likely to be equivalent to the PDA rather than the DDA. In fact, the PDA uses the SSA semantics to perform a reaction firing whereas the DDA does not, and also it also has a policy to avoid creating inconsistencies.

6.2 A DSSA with Delayed Propensity Functions (DPF)

In this section we firstly introduce some notations in order to give a clear exposition of the DSSA with *delayed propensity functions*, shortly denoted as DPF. We assume to consider only systems where all the reactions have got a non-zero delay; the DPF can be easily extended to non-delayed reactions.

We discuss firstly how to compute the putative time for the next reaction to fire. In order to compute the such a value we have to use the propensity functions for the reactions involved in the system. Let us assume to have the system at time t in state $\mathbf{x}$, $\mathbf{X}(t) = \mathbf{x}$, we define, differently from all the DSSAs we previously presented, *delayed propensity functions*, namely propensity functions which depend on past states of the system. Formally, for a generic reaction R_j, its propensity function is evaluated in a state $\mathbf{x}'$ if the system was in state $\mathbf{x}'$ at time $t - \sigma_j$. In the next, such a operation is to be denoted either as $a_j(\mathbf{X}(t - \sigma_j))$ or as $a_j(\mathbf{x}')$ if it is clear that $\mathbf{X}(t - \sigma_j) = \mathbf{x}'$. As a consequence, the summation of all the propensity functions is to be denoted as

$$a_0(t) = \sum_{j=1}^{M} a_j(\mathbf{X}(t - \sigma_j)) \,.$$

In order to be able to define at each step of the algorithm such a quantity, we have to add to the state of our computation the history of the system, as it is to be done in the DDEs. Practically, this means that, in addition to $\mathbf{x}$, we store a set of states

$$\mathbf{H} = \{(t', \mathbf{x}') \mid \mathbf{X}(t') = \mathbf{x}' \wedge t - \sigma^M \leq t' < t\}$$

where $\sigma^M = \max\{\sigma_j \mid j = 1, \ldots, M\}$ is the maximum among all the possible delays, which represents the finite set of *distinct* states in which the system was before reaching $\mathbf{x}$. In order to evaluate a propensity function we have to find in $\mathbf{H}$, for any reaction R_j, the pair $(t_j^*, \mathbf{x}_j^*)$ such that

$$(t_j^*, \mathbf{x}_j^*) = \max\{(t', \mathbf{x}') \in \mathbf{H} \mid t' \leq t - \sigma_j\}$$

so that we are sure that the propensity function is to be evaluated as $a_j(\mathbf{x}_j^*)$ since the system was, at time $t - \sigma_j$, in state $\mathbf{x}_j^*$. This means that, from $\mathbf{H}$, we can evaluate the propensity functions and define their summation $a_0(t)$. Of course, at each step of the algorithm, the set $\mathbf{H}$ is to be updated accordingly to the state-changes in the system, if any.

Now, as in the SSA we define the putative time for the next reaction to fire

$$\tau \sim \mathcal{E}xp\,(a_0(t))$$

where, similarly to the SSA, the exponential random variable as parameter equal to the summation $a_0(t)$. Notice that, in all the algorithms we previously defined such a parameter is the result of evaluating the propensity functions in the current state of the system.

Once that τ has been sampled, we have to check whether it represents a correct time step for the algorithm. Indeed, in the DSSAs we presented a time step was not rejected only if in the time window $[t, t + \tau]$ no reactions were scheduled, and this was independent on the scheduling policy. Such a criterion, as we discussed, is the result of the mathematical foundations of the algorithm, as described by the equations derived in proving the correctness of the DSSAs. A consequence of such a criterion in the SSA is that the quantity $a_0(t) = \sum_{j=1}^{M} a_j(\mathbf{X}(t))$ is constant in the whole time window $[t, t + \tau)$, and this invariant holds also for the DSSAs we previously defined. Consequently,

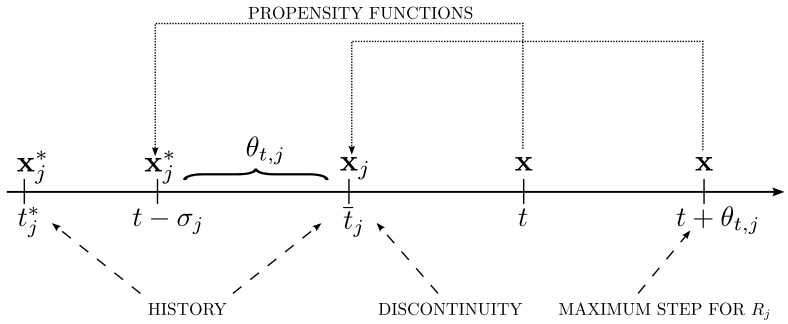

FIGURE 6.1: Maximum step size for a delayed reaction in the DPF.

we expect this invariant to hold here, and in order to define the conditions to have the invariant valid here, we consider for each reaction R_j, the quantity

$$\bar{t}_j = (t - \sigma_j) + \theta_{t,j}$$

with $0 \leq \theta_{t,j} \leq \sigma_j$ as the first time instant immediately subsequent to the time instant $t - \sigma_j$ in which a reaction fired. Intuitively, $\bar{t}_j$ is a pointer to a past-time in which a reaction fired and, hence, there was a state-change. Since state-changes may imply changes in the propensity functions which are state-dependent, then the propensity function of some reactions may change when the time is $t + \theta_{t,j}$. This because, in the DPF, the propensity functions are computed in past-states of the system, hence for reaction R_j its propensity function at time $t + \theta_{t,j}$ depends on the state in which the system was at time $(t - \sigma_j) + \theta_{t,j}$. Of course, not in all the case the propensity functions change, but it most case they do. As a consequence, we define the most general strategy for correctly handling both the situations by considering the worst case, namely when the functions change.

Practically the value $\theta_{t,j}$, which depends on the history of the system, can be again computed by means of $\mathbf{H}$. Indeed, since the pair $(\bar{t}_j, \mathbf{x}_j)$ can be computed as

$$(\bar{t}_j, \mathbf{x}_j) = \min\{(t', \mathbf{x}') \in \mathbf{H} \mid t' > t - \sigma_j\}$$

then we have that $\theta_{t,j} = \bar{t}_j - t + \sigma_j$. If the pair $(\bar{t}_j, \mathbf{x}_j)$ does not exist, namely in $\mathbf{H}$ there are no state-change subsequently to $t - \sigma_j$, then we have $\theta_{t,j} = \infty$. Notice that $\theta_{t,j}$ is a value which represent the maximum value of τ such that the summation $a_0(t)$ does not change in $[t, t+\tau)$. As a consequence, we can think about the time instant $t + \theta_{t,j}$ as generally representing a *discontinuity* which can be formally defined by using the left and right limits

$$\lim_{\tau \to \theta_{t,j}^-} a_0(t + \tau) \neq \lim_{\tau \to \theta_{t,j}^+} a_0(t + \tau)$$

representing the continuity condition for a_0 in $t + \theta_{t,j}$. We remark that this is true only if the propensity functions change around $t + \theta_{t,j}$, as we consider here. In order to have the target invariant, the maximum value for τ is to be chosen as the minimum of all the possible discontinuities, namely as

$$\theta_t = \min\{\theta_{t,j} \mid j = 1, \ldots, M\}.$$

Such a value implies that the quantity $a_0(t)$ is constant in the whole time window $[t, t + \theta_t)$. As a consequence, once τ and θ_t are computed, we can define a policy for if and which reaction is to be fired. In FIGURE 6.1 the scenario we discussed is graphically represented.

In the case that a reaction can fire, $\tau < \theta_t$, its index j is chosen as in the SSA with a probabilistic choice among the values $a_j(\mathbf{x}_j^*)/a_0(t)$. Differently, if no rules can be applied, $\tau \geq \theta_t$, then the algorithm simply moves time forewords (as the other DSSAs did when handling scheduled reactions). Furthermore, whenever a reaction R_j is chosen to fire, the DPF checks its applicability in the same style the PDA did, since it is not guaranteed that the reaction can effectively fire in

Algorithm 4 DSSA DPF (t_0, $\mathbf{x}_0$, T, $\mathbf{H}$)

1: $t \leftarrow t_0$;
2: $\mathbf{x} \leftarrow \mathbf{x}_0$;
3: **while** $t < T$ **do**
4: $a_0(t) \leftarrow \sum_{j=1}^{M} a_j(\mathbf{x}_j^*)$; where $(t_j^*, \mathbf{x}_j^*) = \max(\mathbf{H}(\leq_j))$;
5: let $r_1 \sim U[0,1]$;
6: $\tau \leftarrow a_0(t)^{-1} \ln({r_1}^{-1})$;
7: define $\theta_t = \min\{\min(\mathbf{H}(>_j)) - t + \sigma_j \mid j = 1, \ldots, M\}$;
8: **if** $\tau < \theta_t$ **then**
9: let $r_2 \sim U[0,1]$;
10: let j such that $\sum_{i=1}^{j-1} a_i(\mathbf{x}_i^*) < r_2 \cdot a_0(t) \leq \sum_{i=1}^{j} a_i(\mathbf{x}_i^*)$;
11: **if** $\nu_j^r \prec \mathbf{x}$ **then**
12: $t \leftarrow t + \tau$;
13: $\mathbf{x} \leftarrow \mathbf{x} + \nu_j$;
14: $\mathbf{H} \leftarrow \mathbf{H} \cup \{(t, \mathbf{x})\}$;
15: **end if**
16: **else**
17: $t \leftarrow t + \theta_t$;
18: **end if**
19: **end while**

the current state $\mathbf{x}$. This may seem counterintuitive with respect to the fact that its probability is computed in a past state of the system, however, it is necessary to avoid to create inconsistencies if the reactants needed to fire the reaction are not present in the current state $\mathbf{x}$. Only in the case in which the reaction can be applied the application is performed by using the semantics of the reactions as in the SSA, namely defining $\mathbf{X}(t + \tau) = \mathbf{x} + \nu_j$.

After this considerations, if we denote the two partitions of $\mathbf{H}$ as

$$\mathbf{H}(\leq_j) = \{(t', \mathbf{x}') \in \mathbf{H} \mid t' \leq t - \sigma_j\}$$
$$\mathbf{H}(>_j) = \{(t', \mathbf{x}') \in \mathbf{H} \mid t' > t - \sigma_j\}$$

we can define the DPF as ALGORITHM 4.

The DPF assumes the classical inputs as the other DSSAs plus a *history vector* $\mathbf{H}$ which is defined as

$$\forall t \in [t_0 - \sigma^M, t_0).\ \mathbf{X}(t) = \mathbf{H}(t)\,.$$

Here we use the same notation $\mathbf{H}$ used to denoted the set of past states since such a set is built by the DPF as an extension of the history vector.

The DPF initialize the required structures (steps $(1-2)$) and then, at each iteration, evaluates the delayed propensity functions and generates τ as the SSA does (steps $(4-6)$). In step (7) the closest discontinuity point is computed, such a computation is in general harder than computing the maximum step size in other DSSAs, as we already argued. Steps $(8-13)$ are used to generate the index of the next reaction to fire and, if possible, to fire it. Step (17) simply move forewords time.

A final consideration is worth discussing. It is fairly intuitive that this algorithm is computationally more expensive than the ones we previously discussed. In particular, even if neither the scheduling lists in both the DDA and the PDA, nor the history in the DPF can be apriori bounded, it is clear that the scheduling lists are much less big in space. In fact, in those structures we store a constant information for each entry, while in the history of the DPF we need to store at least a whole copy of the state in which the simulation was.

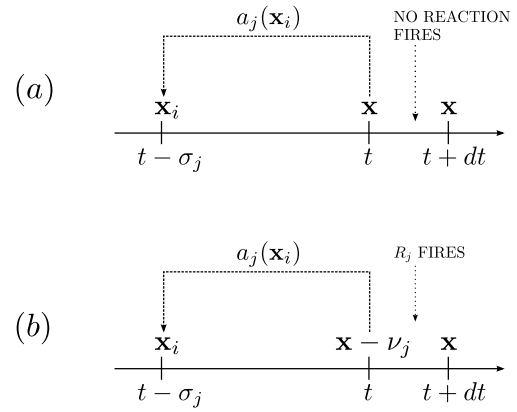

FIGURE 6.2: Events leading to the definition of the DCME for the DPF.

6.3 Mathematical foundations of the DPF

In the next two sections we discuss the mathematical foundations of the DPF. In the first section we show the derivation of a Delayed Chemical Master Equation for target systems of this algorithm. The analytical expression of such an equation turns out to be equal to the one for the PDA (equation (5.1)). In the second section, we build the DPF from rigorous mathematical considerations leading to its correctness and exhibiting its connection with the master equation defined. The combination of these two results leads to the conclusion that the DPF and the PDA are equivalent.

6.3.1 A Delay Chemical Master Equation for the DPF

In this section we derive a Delayed Chemical Master Equation (DCME) for target systems simulated by the DPF. Since we already discussed in detail the derivation of the DCMEs for both the DDA and the PDA, in this section we omit commenting some details.

As in the other equations, we denote as $\mathbb{P}(\mathbf{x}, t \mid I)$ the probability that the system is in state $\mathbf{x}$ at time t given these two facts: the initial configuration $I \equiv \mathbf{x}_0, t_0; \omega$ is $\mathbf{X}(t_0) = \mathbf{x}_0$ and ω is the history function. Analogously as for the CME, we want to define the quantity $\mathbb{P}(\mathbf{x}, t+dt \mid I)$ considering a small enough dt that at most one reaction fires in time $[t, t+dt)$. As in other cases, we consider the following events:

(a) at time t the system is already in state $\mathbf{x}$ and, at delayed time $t-\sigma_j$, the system was in state $\mathbf{x}_i$, no reactions fire;

(b) at time t the system is in state $\mathbf{x}-\nu_j$ and, at delayed time $t-\sigma_j$, the system was in state $\mathbf{x}_i$, reaction R_j fires.

In FIGURE 6.2 the events leading to the definition of the DCME are graphically represented. Notice that these two events are similar to the ones used to define the DCMEs for the algorithms.

Writing up the analytical expressions for the cases, we get

$$\mathbb{P}(\mathbf{x}, t+dt \mid I) = \mathbb{P}(\mathbf{x}, t \mid I) \left(1 - \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j \mid \mathbf{x}, t; I) a_j(\mathbf{x}_i) dt \right) + \sum_{j=1}^{M} \mathbb{P}(\mathbf{x}-\nu_j, t \mid I) \left(\sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t-\sigma_j \mid \mathbf{x}-\nu_j, t; I) a_j(\mathbf{x}_i) dt \right).$$

Similarly to the derivation of the other equations once we apply equation (2.6), divide by dt and

consider to the limit $dt \to 0$ we get the following DCME:

$$\begin{aligned}\frac{\partial \mathbb{P}(\mathbf{x}, t \mid I)}{\partial t} = & - \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x}, t \mid I) a_j(\mathbf{x}_i) \\ & + \sum_{j=1}^{M} \sum_{\mathbf{x}_i \in \mathbb{I}} \mathbb{P}(\mathbf{x}_i, t - \sigma_j; \mathbf{x} - \nu_j^p, t \mid I) a_j(\mathbf{x}_i) \end{aligned} \tag{6.1}$$

which is, as expected, a Partial Delay Differential Equation and is algebraically equivalent to the one ruling the PDA, equation (5.1). This implies that, if the DPF is correct, since it is logically equivalent to this DCME then it is equivalent to the PDA, which is correct. More precisely, this means that once $\mathbf{x}_0$, t_0 and ω are fixed, then $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0, \omega)$ is the same for both the systems. We remark that this does not imply that two single runs of the algorithm will end up in the same state but this means that the average of infinite samples of their traces would be equal.

6.3.2 Correctness of the DPF

In this section we show the correctness of the DPF, as we did for all the other DSSAs. The correctness comes from assumptions an analytical arguments similar to the ones used to derive equation (6.1); this results in the fact that the DPF is logically equivalent to such a DCME and, as a consequence of having the same DCME of the PDA, this implies the DPF and the PDA are equivalent. As we did for other DSSAs, we try to find a form of equation (2.32) in the context of DPF.

As for the SSA, the key in generating simulated trajectories for $\mathbf{X}(t)$ when reaction have delay is not the DCME or even the function $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x}_0, t_0; \omega)$, but rather the new function

$$p(\tau, j \mid \mathbf{x}, t; \mathbf{H})$$

defined such that $p(\tau, j \mid \mathbf{x}, t; \mathbf{H})d\tau$ is the probability, given $\mathbf{X}(t) = \mathbf{x}$ and the history $\mathbf{H}$, that the next reaction in the system occurs in the infinitesimal time interval $[t + \tau, t + \tau + d\tau)$ and is reaction R_j. Notice that, differently from the quantity $p(\tau, j \mid \mathbf{x}, t)$, in this framework we have to consider the whole history of the system. Also, differently from the other DSSAs, we do not consider $p(\tau, j \mid \mathbf{x}, t; \mathbf{\Sigma})$ since here actions fire instantaneously although depending on past states of the system. As expected, $p(\tau, j \mid \mathbf{x}, t; \mathbf{H})$ function is the joint probability density function of two random variables, one modeling the putative time for the next reaction (τ), and the other modeling the choice of the next reaction to fire (j). The next theorem defines an analytical form for such functions and provides the correctness of the DPF.

THEOREM 6.3.1 (DPF Correctness)**.** *The putative time for the next reaction is a continuous random variable τ whose density satisfies*

$$p(\tau \mid \mathbf{x}, t; \mathbf{H}) = \begin{cases} a_0(\mathbf{x}^*) \exp\left(-a_0(\mathbf{x}^*)\tau\right) & \text{if} \quad 0 \le \tau < \theta_t \\ p(\tau - \theta_t \mid \mathbf{x}, t + \theta_t, \mathbf{H}) & \text{if} \quad \tau \ge \theta_t \,. \end{cases} \tag{6.2}$$

when $(t_j^*, \mathbf{x}_j^*) = \max(\mathbf{H}(\le_j)$, $\sum_{j=1}^{M} a_j(\mathbf{x}_j^*) = a_0(\mathbf{x}^*)$, $\theta_t = \min\{\min(\mathbf{H}(>_j)) - t + \sigma_j \mid j = 1, \ldots, M\}$. Moreover, if $\tau < \theta_t$, the index of the next reaction to fire is a discrete random variable j with

$$P(j \mid \tau; \mathbf{x}, t; \mathbf{H}) = \frac{a_j(\mathbf{x}^*)}{a_0(\mathbf{x}^*)} \,. \tag{6.3}$$

Proof. Similarly to equation (2.28), we start by denoting with $P_0(\tau \mid \mathbf{x}, t; \mathbf{H})$ the probability that, given $\mathbf{H}$, no reaction fire in the time interval $[t, t + \tau)$, so that we can write $p(\tau, j \mid \mathbf{x}, t; \mathbf{H})$ as

$$p(\tau, j \mid \mathbf{x}, t; \mathbf{H}) = P_0(\tau \mid \mathbf{x}, t; \mathbf{H}) a_j(\tau) d\tau \tag{6.4}$$

where $a_j(\tau)$ denotes the fact that we evaluate the propensity function for reaction R_j in some state $\mathbf{x}'$ which is the state in which the system was at time $t+\tau-\sigma_j$. Notice that such a state once that $\mathbf{H}$ and τ are known can be fully determined.

Now we are expected to have an analytical expression for $P_0(\tau \mid \mathbf{x}, t; \mathbf{H})$ which turns out to be, in this context, a bit harder than in other DSSAs. Indeed, we can define, similarly as we did for the DCME for the DPF, an ODE for $P_0(\tau \mid \mathbf{x}, t; \mathbf{H})$. Practically, we need to write the the probability $P_0(\tau + d\tau \mid \mathbf{x}, t; \mathbf{H})$ in terms of $P_0(\tau \mid \mathbf{x}, t; \mathbf{H})$ *and* the event "no reactions fire in $[t+\tau, t+\tau+d\tau)$". The probability of a single reaction R_j not to complete in such a time interval is equal to the probability that, in state $\mathbf{x}_j^*$ in which the system was at time $t+\tau-\sigma_j$, either no reactions started or a reaction but R_j started. Technically, such a probability is given by

$$\sum_{\substack{i=1 \\ i\neq j}}^{M} a_i(\mathbf{x}_j^*)d\tau + \left(1 - \sum_{i=1}^{M} a_i(\mathbf{x}_j^*)d\tau\right)$$

which, ones the rightmost summation is split into $\sum_{i=1}^{M} a_i(\mathbf{x}_j^*)$ with $i \neq j$ and $a_j(\mathbf{x}_j^*)$, rewrites as $1 - a_j(\mathbf{x}_j^*)d\tau$. By generalizing this event among all the possible reactions, each of them which could have started at a proper time instant, and since all the starts are independent events, then we have that our target event has probability

$$\prod_{j=1}^{M} \left[1 - a_j(\mathbf{x}_j^*)d\tau\right]$$

where $(t_j^*, \mathbf{x}_j^*) = \max(\mathbf{H}(\leq_j)$. Now, this quantity represents that any of the reaction complete at time $t+\tau$. It is fairly intuitive to notice that the following equation holds

$$\prod_{j=1}^{M} \left[1 - a_j(\mathbf{x}_j^*)d\tau\right] = 1 - \sum_{j=1}^{M} a_j(\mathbf{x}_j^*)d\tau + o(d\tau)$$

where $o(d\tau)$ represents all the terms which are the product of at least two propensity functions. Notice that these terms represent the probability that at least two reaction complete together in $[t+\tau; t+\tau+d\tau)$, after starting in two different time instants in the past. Here comes into play the very same assumption used to derive the DCME for the DPF, hence the fact that the probability that two or more reactions complete in the same $d\tau$ is negligible with respect to the probability of one to complete, and hence we have this approximation

$$\prod_{j=1}^{M} \left[1 - a_j(\mathbf{x}_j^*)d\tau\right] \approx 1 - \sum_{j=1}^{M} a_j(\mathbf{x}_j^*)d\tau\,.$$

By denoting the quantity $\sum_{j=1}^{M} a_j(\mathbf{x}_j^*)$ as $a_0(\tau)$, we can write an equation for $P_0(\tau + d\tau \mid \mathbf{x}, t; \mathbf{H})$ similar to equation (2.4.2)

$$P_0(\tau + d\tau \mid \mathbf{x}, t; \mathbf{H}) = P_0(\tau \mid \mathbf{x}, t; \mathbf{H})(1 - a_0(\tau)d\tau)$$

so that, when considering $d\tau \to 0$, we can get the ODE

$$\frac{dP_0(\tau \mid \mathbf{x}, t; \mathbf{H})}{d\tau} = -P_0(\tau \mid \mathbf{x}, t; \mathbf{H})a_0(\tau)\,. \tag{6.5}$$

This equation models the exponential decay of $P_0(\tau \mid \mathbf{x}, t; \mathbf{H})$, as equation (2.29) was modeling such a decay for $P_0(\tau \mid \mathbf{x}, t)$, and its solution is

$$P_0(\tau \mid \mathbf{x}, t; \mathbf{H}) = P_0(0 \mid \mathbf{x}, t; \mathbf{H}) \exp\left(-\int_0^{\tau} a_0(t')dt'\right)$$

In this case, the initial condition $P_0(0 \mid \mathbf{x}, t; \mathbf{H}) = 1$ since it represents the probability that nothing happens in 0 time whatever $\mathbf{H}$ is, but $-\int_0^\tau a_0(t')dt'$ can not be solved explicitly, and then we have that

$$P_0(\tau \mid \mathbf{x}, t; \mathbf{H}) = \exp\left(-\int_0^\tau a_0(t')dt'\right).$$

Equation (6.4) can be rewritten by means of this last equation

$$p(\tau, j \mid \mathbf{x}, t; \mathbf{H}) = a_j(\tau)d\tau \exp\left(-\int_0^\tau a_0(t')dt'\right)$$

and then can be generalized to get the probability $p(\tau \mid \mathbf{x}, t; \mathbf{H})d\tau$ that a reaction fires at time $t+\tau$

$$p(\tau \mid \mathbf{x}, t; \mathbf{H}) = \sum_{j=1}^{M} p(\tau, j \mid \mathbf{x}, t; \mathbf{H}) = a_0(\tau) \exp\left(-\int_0^\tau a_0(t')dt'\right).$$

This equation is what equation (2.32) corresponds to in the context of the DPF, notice the similarity with equations (4.2) of the DDA and the PDA. As in the DDA and the PDA, we need to understand what equations (2.34) and (2.35) become in this context.

In order to get this result we consider the biggest value φ such that $a_0(t')$ is constant for all the time window $[t, t+\varphi)$ is, in the this context, the smallest of all the possible discontinuities in $\mathbf{H}$, $\varphi = \theta_t = \min\{\min(\mathbf{H}(>_j)) - t + \sigma_j \mid j = 1, \ldots, M\}$. By writing a recursive definition for $a_0(t')$ where $\xi = \sum_{j=1}^{M} a_j(\mathbf{x}_j^*) = a_0(\mathbf{x}^*)$ we have that the two target equations become

$$\int_0^\tau a_0(\mathbf{x}^*)dt' = \log({r_1}^{-1}) \tag{6.6}$$

if $\tau < \theta_t$, and

$$\int_0^{\tau-\theta_t} a_0(t')dt' = \log(cr_2^{-1}). \tag{6.7}$$

if $\tau > \theta_t$ and $\int_0^{\theta_t} a_0(\mathbf{x}^*)dt' = \log(c^{-1})$. As in the other DSSAs we do not consider the case $\tau = \theta_t$ which in the DPF is embedded in equation (6.7). We can do that because of the continuity of the exponential distribution, as in the DDA and the PDA. These two equations give the analytical definition of equation (6.2).

Furthermore, the fact that we generate the index of the next reaction to fire with the same scheme used in the SSA comes from the fact that, if we write the conditional probability $P(j \mid \tau; \mathbf{x}, t; \mathbf{H})$ we have

$$P(j \mid \tau; \mathbf{x}, t; \mathbf{H}) = \frac{p(\tau, j \mid \mathbf{x}, t; \mathbf{H})}{p(\tau \mid \mathbf{x}, t; \mathbf{H})} = \frac{a_j(\mathbf{x}_j^*)\exp\left(-a_0(\mathbf{x}^*)\tau\right)}{a_0(\mathbf{x}_j^*)\exp\left(-a_0(\mathbf{x}^*)\tau\right)} = \frac{a_j(\mathbf{x}^*)}{a_0(\mathbf{x}_j^*)} \tag{6.8}$$

which is the same equation of the SSA rephrased in the context of the DPF when $\tau < \theta_t$. □

By all these considerations it is clear that the choice of generating a value for τ as a sample of an exponential distribution with mean $a_0(\mathbf{x}^*)$, as in step (4) of the DPF, is correct. Also, it is correct if $\tau \geq \theta_t$ simply increase, as in step (17), the clock of the system to $t + \theta_t$, update the state and recompute a sample for an exponentially distributed number with mean given by computing the new values of the propensity functions. Finally, notice that in the recursive formula for the density of τ we wrote $\mathbf{x}$ differently from the PDA where we where able to distinguish among the two possible cases: $\mathbf{x}$ or $\mathbf{x} + \nu_j$. This is correct in the DPF since we know that the state is not changing at time $t + \theta_t$.

Chapter 7

A purely delayed approach with markings

In this chapter we present two new DSSAs, obtained as improvement and combination of the DDA and the PDA.

More precisely, we start discussing a problem of inaccuracy for the naïve PDA presented in CHAPTER 5 and, inspired from that, we define a liveness property for PDA simulations. Secondly, in order to improve the PDA we define a notion of marking for the molecules in the system state. From that we define how to evaluate the propensity functions in presence of markings, how to modify the semantics of firing a reaction with delay and how to update the scheduling list for a system with markings. We end up defining a new version of the PDA named PDA with markings.

Such an algorithm maintains the features of the PDA , and fixes the problem we identified. At the end of the chapter we combine the ideas introduced to define the PDA with markings with the PDA , leading to the definition of a combination of the PDA and the DDA with markings.

7.1 Inaccuracy in the PDA

We proved the naïve PDA to be more suitable than the DDA to model systems in which species involved in a delayed interaction can be involved at the same time in other interactions. We now have to make some considerations about the PDA algorithm we proposed. In particular, there are two scenarios in which the algorithm behavior is not satisfactory, and this the reason it is termed naïve. We go through these scenarios via some examples. For instance, consider a system described by the following initial state and chemical reaction:

$$\mathbf{X}(t_0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad\qquad A \overset{k,\sigma}{\longmapsto} B$$

where the initial state contains one single molecule A and the only reaction is the one transforming a molecule A in a molecule B with a kinetic constant k and a delay $\sigma > 0$. When applying the PDA algorithm to simulate this system it is easy to observe that the algorithm may *over-schedule* the firing of the reaction. Let us assume that t_1 is the time in which the reaction is scheduled, and hence it is assumed to complete at time $t_1 + \sigma$. Moreover, subsequently the system jumps at times t_i, with $i = 2, \ldots, n$ and $t_n < t_1 + \sigma$, performing further scheduling of the same reaction. This could happen because the reaction has a delay and, dependently on the value σ and on the random numbers generated by the algorithm, between time t_0 and $t_1 + \sigma$ the PDA may schedule an arbitrary amount of times the reaction. However, after applying the first scheduled reaction at time $t_1 + \sigma$, the system state becomes the vector

$$\mathbf{X}(t_1 + \sigma) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

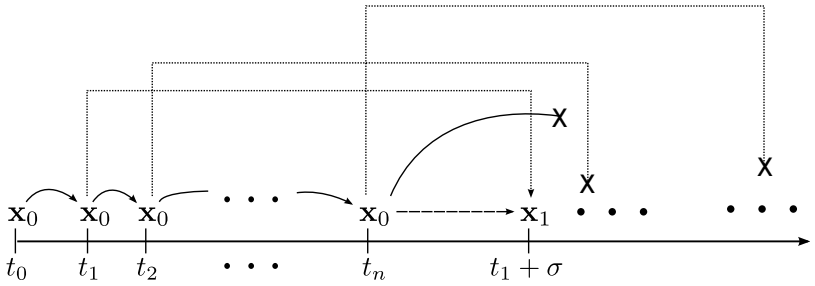

FIGURE 7.1: Over-scheduling in the PDA.

and, consequently, all the other scheduled reactions are not applicable anymore since

$$\nu_1^r = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \not< \mathbf{X}(t_0 + \varphi)\,.$$

In this situation the behavior of the algorithm is technically correct in fact, as expected, just one molecule A is transformed in a molecule B and all the other scheduled events are discarded. However, from the computational point of view this is fairly unsuitable since it schedules a lot of reactions that are never performed. This is a first criticism to the naïve PDA; in Figure 7.1 such scenario is graphically represented.

Now we can define a second and more important criticism to the PDA by building, on top of this scenario, a new model such that the behavior of the PDA becomes incorrect. Imagine, for instance, to have the same scenario enriched with a reaction which produces, by an external unbounded source, molecules of type A

$$\epsilon \overset{k_1,0}{\longmapsto} A\,.$$

As there is no way of *tracking* the time since a molecule is in the system, then there is no way of preventing to apply a scheduled reaction to a molecule A which is not in the current state of the system by at least σ time units. In this enriched scenario it may be the case that molecules A just appearing in the state by the firing of the new reaction, may be used to perform over-scheduled reactions and this is, obviously, incorrect.

Notice that in order to have an incorrect behavior of the PDA it is possible to have even simpler models than the one we discussed here. In fact, such a scenario can theoretically happen also in absence of over-scheduling. Let us discuss this with another example. We consider reaction $A \overset{k,\sigma}{\longmapsto} B$ enriched with both influx and degradation of molecules of type A, namely we have a 3-reactions model with

$$\epsilon \overset{k_1,0}{\longmapsto} A \qquad\qquad A \overset{k_2,0}{\longmapsto} \epsilon\,.$$

Incorrectness of the algorithm can happen with, for instance, the following sequence of steps. The system starts at time t_0 and choses to schedule transformation of A into B, such an event is scheduled at a time $t_1 + \sigma$ where t_1 is the time at which system jumps. Then, the algorithm decides to jump at a time $t_2 < t_1 + \sigma$ and apply degradation. This makes the new state to be

$$\mathbf{X}(t_2) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

and reaction scheduled at time $t_1 + \sigma$ is not aware, in the PDA, of this event. Obviously, we would expect to remove this reaction from the scheduling list since the molecule to which the reaction should be applied is not present anymore in the system state. In fact, in the current state, the

only reaction with positive propensity function is the reaction modeling influx of molecules of type A. Such a reaction can fire at time t_3 which satisfies $t_3 < t_1 + \sigma$, yielding to a new state

$$\mathbf{X}(t_3) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Now, since this state permits the application of scheduled reaction, if the next time for a reaction to fire t_4 is such that $t_4 > t_1 + \sigma$, then the reaction is handled and the system state becomes

$$\mathbf{X}(t_4) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Of course, this is not correct since the molecule A to which the reaction is applied has been created by the previously firing reaction. More precisely, such a molecule is present in the system state for $t_1 + \sigma - t_3$ units of time and, by construction, such a quantity is smaller than σ, which is the time required to complete reaction. We remark that this incorrect behavior is present even in the absence of over-scheduling.

Summarizing, even if the PDA is, for some biological systems, a better candidate than the DDA, it needs to be properly tuned to avoid to simulate incorrect behaviors of the modeled system. The next sections are devoted to the definition of a variant of the PDA facing these issues. The more precise variant of the PDA is also extended to obtain an algorithm that integrates the PDA and the DDA.

Before defining this new algorithm, we spend some more words on the over-scheduling problem in the PDA. Identifying such a problem in the PDA motivated us to investigate an interesting property for PDA simulations, we present such property in the next section.

7.1.1 A liveness property for the PDA

The over-scheduling in the PDA is inspiring for the definition of an analytical property for the PDA.

Proposition 7.1.1. (PDA Liveness) *Reactions scheduled by the* PDA *are* eventually *handled.*

Notice that we do not say that the reaction is performed but simply that it is handled, so it could be not applied. In this sense, this property is a *liveness* property for PDA simulations. In order to prove this property we define the probability of handling the reaction *exactly* after n simulation steps of the algorithm, with $n \in [1, \infty)$. When the system performs n steps we mean that it performs the following sequence of operations:

- it schedule a reaction;
- it performs further $n - 1$ scheduling operations;
- at the n-th step the first of the scheduled reaction is handled.

This scenario is graphically represented in Figure 7.1. Our target property is given by proving that when $n \to \infty$ the probability of the sequence of the first 2 steps becomes 0. Although this property seems quite intuitive since the PDA generates sequences of exponential numbers strictly greater than 1, this property turns out to be quite interesting to be investigated.

We go through its formal definition by firstly defining an analytical form of the probability of a sequence of n steps, in the sense we stated. We assume the system to be in some state $\mathbf{X}(t) = \mathbf{x}$ where no reactions are scheduled, the algorithm starts by choosing a reaction to schedule once that a value for $\tau \sim \mathcal{E}xp(a_0(\mathbf{x}))$ has been chosen. We assume the PDA to schedule a generic delayed reaction R_j, the property we discuss is not related to a particular reaction in the system. In this sense, regardless on the value generated for τ, this step has probability 1. A similar consideration would come from gluing together all the possible system transitions, one for each possible reaction.

Now that the reaction is scheduled at some time $t + \tau + \sigma_j$ the algorithm generates a finite sequence of n exponentially distributed numbers, all with the same mean. If we enumerate the generated values for τ the first $n-1$ numbers are such that

$$\begin{aligned}
&\tau_1 < \sigma_j \\
&\tau_2 < \sigma_j - \tau_1 \\
&\tau_3 < \sigma_j - \tau_1 - \tau_2 \\
&\dots \\
&\tau_{n-1} < \sigma_j - \tau_1 - \tau_2 - \ldots - \tau_{n-2}
\end{aligned}$$

so in general it holds that

$$\forall i = 1, \ldots, n-1.\ \tau_i < \sigma_j - \sum_{k=1}^{i-1} \tau_k \implies \sum_{i=1}^{n-1} \tau_i < \sigma_j$$

if R_j is the reaction scheduled with the first jump of size τ_0. For each of these steps the algorithm also schedules reactions, here we assume that all the scheduled reactions have a delay big enough to be scheduled after time $t + \tau_0 + \sigma_j$. This assumption is perfectly reasonable, let us consider what would happen if it was not true; let us assume that a reaction $R_{j'}$ is scheduled to at time $t' + \sigma_{j'} < t + \tau_0 + \sigma_j$, this reaction is now the next to complete, and it has been scheduled at the step performed at time $t' > t + \tau_0$. It is fairly easy to notice that the property we are considering in the interval $[t + \tau_0, t + \tau_0 + \sigma_j]$ could be analogously defined in the time interval $[t', t + \sigma_{j'}]$ since we are using a distribution for advancing time which has an infinite support.

Now, considering again the sequences of steps of the PDA, the $n-$th step must generate a value for τ such that $\tau_n \geq \sigma_j - \sum_{i=1}^{n-1} \tau_i$ so

$$\sum_{i=1}^{n} \tau_i \geq \sigma_j \,.$$

Let us disregard this last generated number, and let us consider the sequence of the first $(n-1)$ exponential steps. In order to define the probability of this sequence of steps it comes easier to consider each step as described by a random variable with exponential distribution, and all the variables are independent as expected. We denote the set of the n independent exponential variables as

$$\{\tau_i \sim \mathcal{E}xp\,(a_0(\mathbf{x})) \mid i = 1, \ldots, n-1\}\,.$$

Intuitively, the time increment of the algorithm can be defined as a new random variable defined as the summation of all these exponentials, and the probability we are interested in can be rephrased as the probability of such new variable to be greater than σ_j. Formally we have that

$$\left(Y_{n-1} = \sum_{i=1}^{n-1} \tau_i\right) \sim \Gamma(n-1, a_0(\mathbf{x}))$$

so Y_{n-1} follows an Erlang distribution since n is integer and by definition of such distribution we have that

$$\mathbb{P}(Y_{n-1} \leq \sigma_j) = 1 - e^{-a_0(\mathbf{x})\sigma_j} \sum_{i=0}^{n-2} \frac{(a_0(\mathbf{x})\sigma_j)^i}{i!} \tag{7.1}$$

which is the analytical probability of not handling the reaction *within* $(n-1)$ simulation steps of the algorithm. As expected, we have this result which states that eventually the system handles the scheduled reaction since this event has 0 probability

$$\lim_{n\to\infty} \mathbb{P}(Y_{n-1} \leq \sigma_j) = 0$$

which can be easily verified by defining this *power series*, a special case of a *Taylor series* where

$$\lim_{n\to\infty}\left(\sum_{i=0}^{n-1}\frac{(a_0(\mathbf{x})\sigma_j)^i}{i!}\right)=e^{a_0(\mathbf{x})\sigma_j}\,.$$

We can strengthen our conclusion by considering the set

$$\left\{\left(Y_k=\sum_{i=1}^{k}\tau_i\right)\sim\Gamma(k,a_0(\mathbf{x}))\;\middle|\;k\in[1,\infty)\right\}$$

which represents all possible sequences of k steps of the algorithm. The probability of *never* handling the scheduled reaction is given by the limit of the probability of the random variable Y defined as $Y=\sum_{k=1}^{\infty}Y_k$

$$Y\sim\Gamma\left(\sum_{k=1}^{\infty}k,a_0(\mathbf{x})\right)\implies\lim_{k\to\infty}\mathbb{P}(Y\leq\sigma_j)=0$$

A final consideration is worth discussing. A similar property could be defined for the DDA. In there, each of the exponential random variables has a different parameter since the state changes at each time a reaction is scheduled. Moreover. since we focus only on scheduling, then it would be possible to prove that the values of the propensity functions decrease monotonically with the number of scheduling performed. This yields that eventually in the DDA we may use an $\mathcal{E}xp\,(0)$, which has a sample ∞ crossing any possible discontinuity, as required.

7.2 The PDA with markings (MPDA)

In order to get a correct version of the PDA we consider a solution based on a marking of molecules. This variant of the PDA, in the following named *marked Purely Delay Approach* (MPDA), is based on the idea of assigning, to each molecule of the system, a *marking* which permits the identification of the molecules involved in any scheduled reaction. On one side, this will fix the liabilities of the PDA approach but, as it is intuitive, it will be computationally much more expensive than the PDA.

In order to define the MPDA, we assume a slightly simpler framework than the one used to introduce the DDA and the PDA. In particular, we consider only a system in which there are only delayed reactions ($\sigma>0$ for all the reactions) since, as we seen for both the DDA and the PDA, handling non-delayed reactions is done in the same way as in the previously presented algorithms and hence it requires no further study. Also, for the sake of clarity we omit the formal definition of the policy by which scheduled reactions are chosen to be handled since it is the same used in the PDA. In what follows we separately describe the features of the MPDA, formally defined in ALGORITHM 5.

7.2.1 Marking the molecules

The marking of molecules is based on the use of natural numbers as identifiers. In order to get a clear marking policy we classify the molecules of the system. Firstly, the molecules are classified in species, hence a system is described by a set of species $\mathcal{S}=\{\Sigma_1,\ldots,\Sigma_n\}$ which defines the type of molecules we are considering. Furthermore, any Σ_i denotes a set of molecules such that

$$\Sigma_i=\{S^{i,1}_{N_1},\ldots,S^{i,n_i}_{N_{n_i}}\}$$

where $S^{i,j}_N$ is a single molecule belonging to species $\Sigma_i\in\mathcal{S}$, with a unique identifier $j\in\mathbb{N}$ and concurrently performing the reactions in $N\subset\wp(\{z|R_z\in\mathcal{R}\})$, a set of identifiers of the reactions which are present in the current model. Notice that, for any molecule of the model, we carry much

more information than the one we had in the PDA. In particular, for any molecule we can exactly know which reactions it is concurrently performing and, in this context, this means that there is an instance of reaction consuming that molecule which is currently scheduled in the event list. As these sets change during the simulation of a system, we may denote by $\mathcal{S}(t)$ and $\Sigma_i(t)$ the set of species and the set of molecules of species Σ_i at time t, respectively.

The marking of molecules requires to discuss the use of vector $\mathbf{X}(t)$ which, as in the PDA, will be used to observe the state changes due to the reactions firing. The construction of the state vector $\mathbf{X}(t)$ is slightly changed, with respect to the PDA, by the introduction of this marking notation. In particular, we define $\mathbf{X}(t)$ as

$$\mathbf{X}(t) = (|\Sigma_1(t)|, \ldots, |\Sigma_n(t)|) \tag{7.2}$$

where $|\Sigma_i(t)|$ denotes the cardinality of the set $\Sigma_i(t)$. Notice that $|\Sigma_i(t)|$ represents the number of molecules of species Σ_i at time t, exactly as the element $X_i(t)$ of $\mathbf{X}(t)$ in the definition of the PDA.

As regards $\mathbf{X}(t_0)$, given an initial input state $\mathbf{x}_0$ for the mPDA, a proper initial marking for the system to simulate has to be computed. Let us assume $\mathbf{x}_0 = (X_1(t_0), \ldots, X_n(t_0))$, the set of species can be defined as $\mathcal{S}(t_0) = \{\Sigma_1(t_0), \ldots, \Sigma_n(t_0)\}$ where

$$\forall i = 1, \ldots, n.\ \Sigma_i(t_0) = \{S_\emptyset^{i,1}, \ldots, S_\emptyset^{i,X_i(t_0)}\}. \tag{7.3}$$

This construction creates n sets of species types $\Sigma_i(t_0)$ and, for each of them, it creates $X_i(t_0)$ molecules, each one with a different identifier, which are performing no reactions (at time t_0 the event list is empty, hence their set subscript is the empty set). This guarantees that the molecules are correctly marked and, hence, distinguishable.

In general, at any step of computation, the mPDA algorithm may modify some of the sets $\Sigma_i(t) \in \mathcal{S}(t)$. The discussion about how the mPDA modifies the sets in $\mathcal{S}(t)$, accordingly to the time-evolution of the simulated system, is presented in the next subsections.

7.2.2 Evaluating the propensity functions

The firing of a reaction is the result of a probabilistic choice based on the propensity function of the reaction, evaluated in the current state of the simulation. In order to explain how the mPDA works, we recall the assumption which is at the basis of the definition of this PDA algorithm. The molecules can perform multiple reactions in parallel, but each molecule can be involved in each reaction at most once at a time. The first reaction to finish interrupts the others running in parallel and involving the same molecule. In order to avoid over-scheduling phenomena it is to be ensured that propensity function of a reaction depends only on the occurrences of reactants that are not yet involved in the same reaction.

Let us assume that we have to evaluate the propensity function of a reaction $R_z : M \xrightarrow{k,\sigma} P \in \mathcal{R}$ such that R_z transforms a multiset of molecules M in a multiset of molecules P with a kinetic constant $k \in \mathbb{R}$. More precisely, let us assume M to be a multiset of the form $\{(1, n_1), \ldots, (w, n_w)\}$, namely reaction R_z transforms, for any $j = 1, \ldots, w$, a number of n_j molecules of species Σ_j. Notice that this corresponds to a set representation of the state–change vector for reactants.

As we want to take into consideration only the molecules in the current state of the simulation which are not already involved in any scheduled firing of reaction R_z, then we have to filter those that are candidate for being used, if any. Let us denote by $[\Sigma_i(t), z]$ the set of identifiers of molecules belonging to species $\Sigma_i(t)$ which have to be considered in the evaluation of the propensity function of R_z, namely the set

$$[\Sigma_i(t), z] = \{j \mid S_N^{i,j} \in \Sigma_i(t) \wedge z \notin N\}.$$

Notice that this set is obtained by considering all the molecules in the current system, and by filtering them on the basis of the marking information that the mPDA stores in $\mathcal{S}(t)$. In the PDA this set could not have been defined.

Given $\mathbf{X}(t) = \mathbf{x}$, the propensity function $a_z(\mathbf{x})$ must consider only those molecules required by M which are not already performing reaction R_z, hence it can be defined as follows:

$$a_z(\mathbf{x}) = k \cdot \prod_{(i,n_i)\in M} \binom{|[\Sigma_i(t), z]|}{n_i} \tag{7.4}$$

where $|[\Sigma_i(t), z]|$ denotes the cardinality of the set $[\Sigma_i(t), z]$. This definition of the function $a_z(\mathbf{x})$ is such that MPDA propensity functions compute, in general, strictly smaller values than the PDA ones. Again, the PDA cannot distinguish the molecules which are performing a reaction from those which are not.

7.2.3 Scheduling a reaction to fire

Whenever the propensity functions have been evaluated, for any $R_z \in \mathcal{R}$, accordingly to the definition (7.4), the index of the reaction to fire can be chosen with the same policy used in the PDA. However, having a marking of molecules, the MPDA has a further level of choice to determine to which molecules the reaction will be applied.

To clarify this, as an example consider a system with two distinct molecules of the same type and both available for being consumed by a reaction. Whenever the MPDA decides to fire that reaction, it has to choose to which of the two molecules the reaction will be applied. This further choice is required by the MPDA because it stores individual information about the molecules and, hence, there exist two different destination markings that the system may reach. Notice that, as the PDA abstracted these informations, it did not perform this further choice.

In order to define this further probabilistic choice, assume the MPDA has chosen to schedule the firing of the reaction $R_z : \{(1, n_1), \ldots, (w, n_w)\} \xrightarrow{k,\sigma} P$ introduced in the previous section. The MPDA stores in the event list the same information of the PDA , namely the index of the reaction, z, and the time in which it will fire, some $t + \tau + \sigma$ if t is the current time, τ is the putative time for next reaction as computed in the PDA and σ is the delay of the reaction. Together with this information, the MPDA stores in each element of the event list a set of labels E representing the identifiers of the molecules which will be consumed by the reaction, when handled. The set E contains pairs of natural numbers and is such that if $(i, j) \in E$ then the molecule $S_N^{i,j} \in \Sigma_i(t)$ is involved in the reaction. The set E is built by considering the molecules which can effectively perform reaction R_z. Formally, for all $(i, n_i) \in M$, we choose n_i molecules from the set $[\Sigma_i, z]$. Each molecule is chosen with probability $|[\Sigma_i(t), z]|^{-1}$, hence the probability of choosing a set E, with a system at time t, denoted by $P(E, t)$, is defined as

$$P(E, t) = \prod_{(i,n_i)\in M} \binom{|[\Sigma_i(t), z]|}{n_i}^{-1}. \tag{7.5}$$

The MPDA updates the system clock to a value $t + \tau$, stores the triple $(z, t + \tau + \sigma, E)$ in the event list and changes the marking of the molecules belonging to the set E. The marking is updated to store the information that the molecules in E are performing reaction R_z. This will guarantee that, when evaluating the propensity function for reaction R_z in the next time, the molecules in E will not be counted again, as expected. The updated set $\mathcal{S}(t + \tau)$, built by modification of the set $\mathcal{S}(t)$ satisfies the following proposition

$$\forall i = 1, \ldots, n. \Sigma_i(t + \tau) = \{S_N^{i,j} \in \Sigma_i(t) \mid (i, j) \notin E\} \cup \{S_{N\cup\{z\}}^{i,j} \in \Sigma_i(t) \mid S_N^{i,j} \in \Sigma_i(t) \wedge (i, j) \in E\}. \tag{7.6}$$

Intuitively, any molecule in $\Sigma_i(t)$ that has not been assigned to the firing of reaction R_z is simply copied in $\Sigma_i(t + \tau)$. Differently, all the molecules assigned to this firing of R_z, are copied in $\Sigma_i(t + \tau)$ with the index z added to their set of concurrently running reactions.

7.2.4 Handling a scheduled reaction

When the MPDA decides, with the system at time t, to handle a scheduled reaction R_z it finds, as information in the event list, a triple (z, t', E) where z is the identifier of the reaction to fire, t' is the time to which the clock must be set and E is the set of identifiers of the molecules which will be consumed by the reaction. It is guaranteed, by construction, that the molecules denoted by the set E are still present in the current state of the simulation. Hence, differently from the PDA , the condition $\nu_z^r \prec \mathbf{x}$ has not to be checked at this time.

The scheduled reaction is applied, as expected, by using the same policy of the PDA, namely the reactants are removed and the products are inserted. However, the MPDA must perform some additional operations to keep the marking of the molecules correct.

First of all, let us assume the set $E = \{(s_1, l_1), \ldots, (s_m, l_m)\}$, then all the molecules denoted by these labels in $\mathcal{S}(t)$ must not be present anymore in the set $\mathcal{S}(t')$, built by the MPDA to represent the markings after the application of reaction R_k. In particular, for any $j = 1, \ldots, m$, the molecule $S_N^{s_j,l_j}$ must be removed from the proper set in $\mathcal{S}(t')$. To define this, we start by defining the following sets

$$\forall i = 1, \ldots, n.\ \Sigma_i(t') = \Sigma_i(t) \setminus \{S_N^{i,j} \in \Sigma_i(t) \ \mid \ (i,j) \in E\} \tag{7.7}$$

Notice that this corresponds to remove exactly the number of reactants required by the application of the reaction R_z. Consequently, given the state vector $\mathbf{X}(t) = \mathbf{x}$ defined accordingly to (7.2), this new marking corresponds to a new state $\mathbf{x} - \nu_z^r$.

As regards the interruption of the concurrently running reactions which were assuming to use the reactants just consumed by reaction R_z, the MPDA performs two operations. Firstly, the MPDA interrupts these reactions by removing them from the event list and, secondly, it unlocks all the involved partners molecules, so that they may start again, in the future, the interrupted reactions.

The interruption of the scheduled reactions is trivial. Let us denote with $\mathcal{E}(t)$ the event list of the system at time t, all the reactions to be interrupted are those which contain, at least, one reactant which is consumed by reaction R_z. We denote by $\mathcal{B}(t)$ the set of reactions to be interrupted at time t, namely the set

$$\mathcal{B}(t) = \{(\tilde{w}, \tilde{t}, \tilde{E}) \in \mathcal{E}(t) \mid \tilde{E} \cap E \neq \emptyset\}.$$

Consequently, the MPDA modifies the event list $\mathcal{E}(t)$ creating a new event list $\mathcal{E}(t')$ such that

$$\mathcal{E}(t') = \mathcal{E}(t) \setminus \mathcal{B}(t). \tag{7.8}$$

Unlocking the partners of the interrupted reactions is less easy. First of all, when considering a generic molecule $S_N^{i,j} \in \Sigma_i(t')$, where $\Sigma_i(t')$ is a set of molecules satisfying (7.7), it may be the case that it is coupled to some of the events which have been interrupted in $\mathcal{E}(t')$, and these events belong to the set $\mathcal{B}(t)$. Also, it may be the case that the molecule is performing other reactions which have not been interrupted. In general, even if $w \in \{w | (w, t, E) \in \mathcal{B}(t)\}$, this does not imply that all the scheduled events referring to reaction R_w have to be interrupted. Clearly, this depends on the one-to-many correspondence between a reaction and all the related scheduled events. Hence, in order to filter the reactions which have been really interrupted for a molecule $S_N^{i,j} \in \Sigma_i(t')$, we define the set

$$\mathcal{D}(t, i, j) = \{w \ \mid \ (w, \tilde{t}, \tilde{E} \cup \{(i,j)\}) \in \mathcal{B}(t)\}.$$

The construction of the set $\mathcal{D}(t, i, j)$ is straightforward. All the reactions which have to be interrupted, with respect to molecule $S_N^{i,j}$, are only those relative to events effectively interrupted and such that the molecule was assumed to be consumed by that instance of reaction. This constraint filters any possible collision between the indexes of the reactions relative to the interrupted events and those which are performed with partners whose are not affected by the application of the scheduled reaction R_z.

After this considerations, we can formally define how the interruption of some events affects the marking of the molecules by defining these new sets

$$\forall i = 1, \ldots, n.\ \Sigma_i'(t') = \{S_{N'}^{i,j} \ \mid \ S_N^{i,j} \in \Sigma_i(t') \ \wedge \ N' = N \setminus \mathcal{D}(t, i, j)\}. \tag{7.9}$$

Algorithm 5 DSSA MPDA (t_0, $\mathbf{x}_0$, T)

$t \leftarrow t_0$;
build the initial marking w.r.t definition (7.3) by using $\mathbf{x}_0$;
while $t < T$ **do**
  $a_0(\mathbf{x}) \leftarrow \sum_{j=1}^{M} a_j(\mathbf{x})$ where $a_j(\mathbf{x})$ is evaluated w.r.t. definition (7.4);x
  let r_1, $r_2 \sim U[0,1]$;
  $\tau \leftarrow a_0(\mathbf{x})^{-1} \ln({r_1}^{-1})$;
  let j such that $\sum_{i=1}^{j-1} a_i(\mathbf{x}) < r_2 \cdot a_0(\mathbf{x}) \leq \sum_{i=1}^{j} a_i(\mathbf{x})$;
  if $\exists R_k \in \mathcal{R}$ scheduled as the first to complete at $t + \tau_k \wedge \tau_k < \tau$ **then**
    update the event list w.r.t. definition (7.8);
    update the marking w.r.t definitions (7.7), (7.9), (7.10) and (7.11);
    $t \leftarrow t + \tau_k$;
  else
    choose, w.r.t. definition (7.5), the set of reactants E used by reaction R_j;
    update the marking w.r.t. definition (7.6);
    schedule the triple $(j, t + \sigma_j + \tau, E)$;
    $t \leftarrow t + \tau$;
  end if
end while

Notice that, as this definition does not modify the number of molecules present in the markings, then this marking, with respect to definition (7.2), still represents the vector state $\mathbf{x} - \nu_z^r$.

Finally, we discuss how the insertion of the products affects the marking of the molecules in the system, with respect to the sets just created. Let us assume that the scheduled reaction R_z creates a multiset of products $P = \{(1, n_1), \ldots, (p, n_p)\}$, namely R_z produces, for any $j = 1, \ldots, p$, a number n_j of new molecules of species Σ_j.

The creation of new objects to add to the sets $\Sigma'_i(t')$ requires to assign them new fresh identifiers respecting the uniqueness of the markings. As the marking is based on the use of natural numbers, the MPDA has an infinite set of numbers from which to choose the new identifiers. Let us denote, for a species Σ_i, the maximum among all the used identifiers appearing in $\Sigma'_i(t')$ as follows

$$\mu_i = \max\{j \mid S_N^{i,j} \in \Sigma'_i(t')\}.$$

Hence, for the set $\Sigma'_i(t')$, the creation of n_i non colliding identifiers can be obtained by choosing the n_i successors of the number μ_i. By these consideration we can define the following sets

$$\forall i = 1, \ldots, n.\ \Sigma''_i(t') = \Sigma'_i(t') \cup \{S_\emptyset^{i,\mu_i+1}, \ldots, S_\emptyset^{i,\mu_i+n_i} \mid (i, n_i) \in P\}. \tag{7.10}$$

Finally, the complete marking computed by the MPDA after the application of a scheduling rule is defined as

$$\mathcal{S}(t') = \{\Sigma''_1(t'), \ldots, \Sigma''_n(t')\}. \tag{7.11}$$

This new marking is obtained by modifying the one representing, accordingly to definition (7.2), the state vector $\mathbf{x} - \nu_z^R$. As this marking is built by inserting, for each species, exactly the number of product molecules of reaction R_z, then this new marking corresponds to the state vector $\mathbf{x} - \nu_z^r + \nu_z^p = \mathbf{x} + \nu_z$ which is, as expected, the resulting state of the correct application of reaction R_z.

An example computation. In this section we show an example MPDA computation to clarify the approach. Let us consider a system similar to the one we discussed in the beginning of this chapter. We have two reactions

$$R_1 : A \xmapsto{k_1,\sigma_1} B \qquad\qquad R_2 : A \xmapsto{k_2,\sigma_2} C$$

so that molecules of species A should be able to start both the reactions. We consider a simple initial state

$$\mathbf{X}(t_0) = (3, 0, 0)^T .$$

The initial marking $\mathcal{S}(t_0)$ is defined as $\mathcal{S} = \{\Sigma_1(t_0), \Sigma_2(t_0), \Sigma_3(t_0)\}$ where Σ_1, Σ_2 and Σ_3 denote molecules of type A, B and C. These sets are defined as

$$\Sigma_1(t_0) = \{A^1_\emptyset, A^2_\emptyset, A^3_\emptyset\} \qquad \Sigma_2(t_0) = \Sigma_3(t_0) = \emptyset$$

For clarity, here we shortened the notation denoting with $A^1_\emptyset$ the molecule $S^{1,1}_\emptyset$. Once these sets are defined the simulation can start, and firstly propensity functions are evaluated. In this sense, the set $[\Sigma_1(t_0), 1]$ and $[\Sigma_1(t_0), 2]$ are defined as

$$[\Sigma_1(t_0), 1] = [\Sigma_1(t_0), 2] = \{1, 2, 3\}$$

which represents the fact that in the initial marking no molecules A are performing any reaction. Accordingly to our definitions, whatever reaction is chosen, each of the molecules A is chosen with $1/3$ probability. Reactions are chosen with probability $3k_1/3(k_1 + k_2)$ and $3k_2/3(k_1 + k_2)$, respectively. Let us assume to chose molecule marked with 1 to fire reaction R_1, the marking for $\Sigma_1(t_1)$ is defined as

$$\Sigma_1(t_1) = \{A^1_{\{1\}}, A^2_\emptyset, A^3_\emptyset\}$$

where time t_1 includes the τ generated by the MPDA. For the sake of clarity, we only discuss marking features of the algorithm. Now, a reaction is scheduled, and at the next step we have that the sets $[\Sigma_1(t_1), 1]$ and $[\Sigma_1(t_1), 2]$ change as follows

$$[\Sigma_1(t_0), 1] = \{2, 3\} \qquad [\Sigma_1(t_0), 2] = \{1, 2, 3\}$$

to reflect that there is 1 molecule A currently performing the scheduled reaction. This makes the reactions to be chosen with probability $2k_1/(2k_1 + 3k_2)$ and $3k_2/(2k_1 + 3k_2)$, respectively. Let us assume to chose molecule marked with 3 to fire reaction R_2, the marking for $\Sigma_1(t_2)$ is defined as

$$\Sigma_1(t_2) = \{A^1_{\{1\}}, A^2_\emptyset, A^3_{\{2\}}\} .$$

Now, let us assume to fire again reactions R_1 and R_2 in this order, and to pick molecules marked with 3 and 2. Then we have a new marking at time t_3

$$\Sigma_1(t_3) = \{A^1_{\{1\}}, A^2_{\{2\}}, A^3_{\{1,2\}}\} .$$

Now, let us assume that the MPDA decides to handle a scheduled event, the first to complete is the first firing of reaction R_1, which correctly completes without interrupting any running reactions. We have the new marking to be

$$\Sigma_1(t_4) = \{A^2_{\{2\}}, A^3_{\{1,2\}}\} \qquad \Sigma_2(t_4) = \{B^1_\emptyset\} \qquad \Sigma_3(t_4) = \emptyset .$$

If the reaction to complete is now the second instance of the reaction R_1 running, then we have a new marking with

$$\Sigma_1(t_5) = \{A^2_{\{2\}}\} \qquad \Sigma_2(t_5) = \{B^1_\emptyset, B^2_\emptyset\} \qquad \Sigma_3(t_5) = \emptyset .$$

From the scheduling list the instance of reaction R_2 which was running and involving molecule A with identifier 3 has been correctly interrupted. Similar considerations could be done for the completion or the firing of other reactions in the current marking.

7.3 Combining the mPDa and the DDa

In this section we define a stochastic simulation algorithm which combines the delay as duration approach and the purely delayed approach in its most precise definition. This will allow biological phenomena that cannot be suitably dealt with by only one of the two approaches, to be studied.

The framework in which we define this DSSa, in the following denoted as Full DSSa, is a simple modification of the one in which we defined the mPDa. This requires to redefine the DSSa with DDa in a framework were markings are present. As regards the notation, we introduce two disjoint sets of possible reactions $\mathcal{R} = \mathcal{R}_\mathcal{D} \cup \mathcal{R}_\mathcal{P}$ where $\mathcal{R}_\mathcal{D}$ and $\mathcal{R}_\mathcal{P}$ are the sets of reactions that are treated with a DDa approach and a mPDa approach, respectively.

In what follows we describe the Full DSSa, whose formal definition is given in Algorithm 6.

Marking the molecules

Marking the molecules is necessary to use the mPDa inside the Full DSSa. Clearly, the marking defined by the mPDa in Section 7.2.1, together with definitions (7.2) and (7.3), is still valid in the Full DSSa.

Evaluating the propensity functions

We introduced two disjoints sets of reactions, $\mathcal{R}_\mathcal{D}$ and $\mathcal{R}_\mathcal{P}$, in order to separate reactions whose delays have to be considered as durations from those whose delays are pure. However, it is easy to notice that, for any reaction in $R_z \in \mathcal{R}$, its propensity function can be correctly defined as in (7.4). This can be done because the different interpretations of the delays do not require different definitions of the propensity functions, but simply different semantics of the firings of reactions.

Despite this similarity, it is worth making a simple consideration about reactions in $\mathcal{R}_\mathcal{D}$. Those reactions are such that, whenever started, they remove the reactants from the state of the simulation and, when the firing terminates, they add to the state their products. Hence, if a reaction of set $\mathcal{R}_\mathcal{D}$ is performed by a molecule $S_N^{i,j} \in \Sigma_i(t)$, then all the reactions concurrently running in N have to be interrupted and, the involved partners, have to be unlocked. By this consideration it is easy to notice that $\forall i \in N.R_i \in \mathcal{R}_\mathcal{P}$ and, hence, it holds that $\forall t > t_0.\ \forall z \in \mathcal{R}_\mathcal{D}.\ [\Sigma_i(t), z] = \Sigma_i(t)$. Summarizing, evaluating the propensity function of a reaction from set $\mathcal{R}_\mathcal{D}$ does not require to define the set $[\Sigma_i(t), z]$, an operation whose cost is at most linear in the size of $\Sigma_i(t)$, and, consequently, it is computationally less expensive than the evaluation of a propensity function of a reaction in $\mathcal{R}_\mathcal{P}$.

Scheduling a reaction to fire

Reactions in $\mathcal{R}_\mathcal{P}$ are scheduled accordingly to the definitions (7.5) and (7.6) whereas, the reactions in the set $\mathcal{R}_\mathcal{D}$ are scheduled with a different policy.

Assume that the Full DSSa wants to schedule a reaction $R_w \in \mathcal{R}_D$ at time $t + \tau + \sigma_w$. Firstly, the Full DSSa must choose the reactants to which the reaction is applied. As this choice is independent with respect to the interpretation of the delays, the set E to which the reaction will be applied can be chosen accordingly to definition (7.5), as in the mPDa.

Now, as the state must be modified by the removal of the reactants, the Full DSSa changes the marking accordingly to definition (7.7) which corresponds exactly to this operation. Furthermore, as the Full DSSa has, for all molecules in the set E, to interrupt all the reactions that they are concurrently performing, it modifies the event list accordingly to definition (7.8). Finally, the Full DSSa further modifies the marking accordingly to definition (7.9) in order to unlock the partners involved in the interrupted reactions.

The scheduling of the reaction is then performed by adding, to the event list $\mathcal{E}(t)$, a pair $(w, t + \tau + \sigma_w)$. Consequently, the event list in the case of the Full DSSa contains some triples referring to scheduled reactions belonging to set $\mathcal{R}_\mathcal{P}$, and some pairs referring to scheduled reactions belonging to set $\mathcal{R}_\mathcal{D}$.

Algorithm 6 FULL DSSA(t_0, $\mathbf{x}_0$, T)

```
1: t ← t_0;
2: build the initial marking w.r.t definition (7.3) by using x_0;
3: while t < T do
4:    a_0(x) ← Σ_{j=1}^{M} a_j(x) where a_j(x) is evaluated w.r.t. definition (7.4);x
5:    let r_1, r_2 ~ U[0,1];
6:    τ ← a_0(x)^{-1} ln(r_1^{-1});
7:    let j such that Σ_{i=1}^{j-1} a_i(x) < r_2 · a_0(x) ≤ Σ_{i=1}^{j} a_i(x);
8:    if ∃R_k ∈ R scheduled as the first to complete at t + τ_k ∧ τ_k < τ then
9:       t ← t + τ_k;
10:      if R_k ∈ R_D then
11:         update the marking w.r.t definitions (7.10) and (7.11);
12:      else
13:         update the event list w.r.t. definition (7.8);
14:         update the marking w.r.t definitions (7.7), (7.9), (7.10) and (7.11);
15:      end if
16:   else
17:      choose, w.r.t. definition (7.5), the set of reactants E used by reaction R_j;
18:      t ← t + τ;
19:      if R_k ∈ R_D then
20:         update the event list w.r.t. definition (7.8)'
21:         update the marking w.r.t. definitions (7.7) and (7.9);
22:         schedule the pair (j, t + σ_j + τ);
23:      else
24:         update the marking w.r.t. definition (7.6);
25:         schedule the triple (j, t + σ_j + τ, E);
26:      end if
27:   end if
28: end while
```

We remark that, in the MPDA, definitions (7.7), (7.8) and (7.9) were introduced when handling a scheduled reaction. The fact that for a reaction in $\mathcal{R}_\mathcal{D}$ the FULL DSSA uses these definitions at the time of scheduling the reaction is due to the different interpretations of delays.

Handling a scheduled reaction

Scheduled reactions belonging to set $\mathcal{R}_\mathcal{P}$ are handled, accordingly to the MPDA, as explained in SECTION 7.2.4.

Differently, handling a reaction with a DDA approach is trivial because the major computational effort has been done when it was scheduled. Assume that the FULL DSSA wants to handle a scheduled reaction described by the pair (w, t') where $R_w \in \mathcal{R}_\mathcal{D}$. In order to insert the product molecules of R_w by modifying the current marking the FULL DSSA modifies $\mathcal{S}(t)$ by applying definitions (7.10) and (7.11).

Part II

Modeling Biological Systems with Delays

Chapter 8

CCS with delayed actions

The aim of this chapter is to define the core of more complex algebras, based on the *Calculus of Concurrent Systems* (CCS) (Milner, 1980), which would like to support actions with delays following either the delay-as-duration or the purely delayed approach or, more generally, actions with duration ruled by a general distribution.

We start by recalling a definition of CCS, and then we discuss how to extend such a process algebra with delayed actions. This yields to the definition of algebras where actions are non-instantaneous since delays are such that the start and the completion of an action are two detached events. In the original CCS framework actions are instantaneous.

The reason for choosing CCS as a base process algebra is that it contains much of the intuition of more complex process algebras based on dyadic communication, and hence addressing these issues in this calculus should be similar to addressing the same issues in languages extending CCS. Moreover, in the next chapter we focus on a more complex stochastic process algebra based on CSP-style communication. Facing both the CCS-based and the CSP-based approaches should give an exhaustive introduction on formal modeling biological systems with delays.

In defining semantics of CCS with actions with delays, for the sake of simplicity, we do not take into consideration quantitative time and stochastic aspects. The result is that, even with this simplifications the semantics we define are non trivial. However, we are able to show that classical results on equivalences for processes are still valid in CCS where actions follow either the delay-as-duration or the purely delayed approach.

8.1 The CCS: syntax, semantics and equivalences

In this section we recall the syntax of CCS processes. When applying CCS to the modeling of biological systems we adopt a paradigm sometimes referred to as *processes-as-molecules*, meaning that in the modeling of a system we associate a single process with a molecule and a binary reaction with a binary synchronization between processes. Consequently, to a synchronization between two processes the firing of an action in the algebra corresponds.

Let us assume the following infinite sets of actions

$$Act = \{\alpha, \beta, \ldots\} \qquad Act_\tau = Act \cup \{\tau\}$$

and a function $\overline{\ }: Act \to Act$ such that $\overline{\overline{\alpha}} = \alpha$. As always, Act_τ denotes the set of actions enriched with the special internal action τ. Let us denote an infinite set of names as

$$\mathcal{N} = \{A, B, \ldots\}$$

the abstract syntax of CCS is defined as follows.

Definition (CCS Processes) *Processes* of CCS are defined by the following grammar:

$$P \;:=\; 0 \;\mid\; \alpha.P \;\mid\; P + P \;\mid\; P \mid P \;\mid\; A$$

$$\texttt{(Prefix)} \quad \alpha.P \xrightarrow{\alpha} P$$

$$\texttt{(Choice}_1\texttt{)} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\texttt{(Choice}_2\texttt{)} \quad \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

$$\texttt{(Constant)} \quad \frac{P \xrightarrow{\alpha} P' \qquad A \stackrel{def}{=} P}{A \xrightarrow{\alpha} P'}$$

$$\texttt{(Coop}_1\texttt{)} \quad \frac{P \xrightarrow{\alpha} P' \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

$$\texttt{(Coop}_2\texttt{)} \quad \frac{Q \xrightarrow{\alpha} Q' \quad \alpha \in Act_\tau}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\texttt{(Coop}_3\texttt{)} \quad \frac{P \xrightarrow{\alpha} P' \qquad P \xrightarrow{\overline{\alpha}} P'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

TABLE 8.1: The relation $\rightarrow \subseteq \mathcal{P} \times Act_\tau \times \mathcal{P}$ for CCS.

where $\alpha \in Act$ and $A \in \mathcal{N}$. We denote the set of all processes by $\mathcal{P}$.

As usual, 0 denotes the classical idle process that can perform no action. Process $\alpha.P$ can perform action α becoming P. A process $P + Q$ is able to start actions of both P or Q. In the classical semantics of CCS, if an action α in P (resp. Q) starts then Q (resp. P) is discarded. A process $P \mid Q$ is the parallel composition of processes P and Q. If in $P \mid Q$ process P performs α and Q performs $\overline{\alpha}$, then P and Q synchronize and $P \mid Q$ exhibits the internal action τ. Finally, constants A are used to specify recursive systems. In general, systems are specified as a set of constant defining equations of the form $A \stackrel{def}{=} P$.

We introduce some notions on syntax of CCS processes.

Definition (Guarded Process) In a sum $\alpha_i.P'_i + \ldots + \alpha_n.P'_n$ we say that P'_i *is guarded by* α_i, and that the whole sum is guarded. Moreover, a process $P \mid Q$ is guarded if P and Q are guarded. Finally $A \stackrel{def}{=} P$ is guarded if P does.

Such processes are such that in a summation $\alpha_i.P'_i + \ldots + \alpha_n.P'_n$ the action α_i must happen before P_i becomes active. Among all the possible processes which can be generated by CCS syntax we restrict to considering only closed and guarded processes, see SECTION 2.5.

For CCS a Structural Operational Semantics (SOS) (Plotkin, 1981) can be given by means of relation $\rightarrow \subseteq \mathcal{P} \times Act_\tau \times \mathcal{P}$ whose definition is given in TABLE 8.1.

Definition (CCS Semantics) The *semantics* of CCS is given by the LTS $(\mathcal{P}, Act_\tau, \rightarrow)$ where $\rightarrow$ is the minimal relation satisfying the rules given in TABLE 8.1.

Notice that timing aspects in this semantics are not considered, as we already said. Before showing an example CCS model we recall an important result for CCS. Along with the study of the semantics and the expressiveness of formal languages a lot of study has been devoted to the definition of formal equivalences for processes such as bisimulation. Such a relation is a central notion in concurrency theory and the original definition of the bisimulation relation has been recalled in SECTION 2.5. We contextualize such a definition in CCS as follows.

Definition (CCS Bisimulation) Given a LTS $(\mathcal{P}, Act_\tau, \rightarrow)$ a binary relation $R \subseteq \mathcal{P} \times \mathcal{P}$ is a

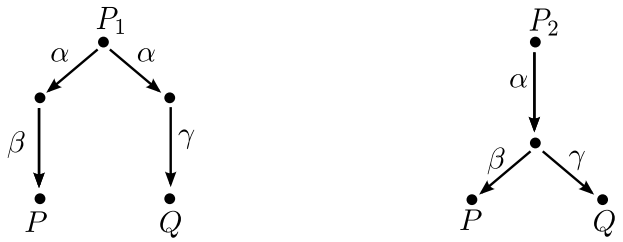

FIGURE 8.1: The LTSs for the two non bisimilar CCS processes.

bisimulation if, for any $(P, Q) \in R$, the following propositions hold:

$$\forall P' \in \mathcal{P}, \alpha \in Act_\tau . P \xrightarrow{\alpha} P' \implies \exists Q' \in \mathcal{P} . Q \xrightarrow{\alpha} Q' \wedge (P', Q') \in R$$
$$\forall Q' \in \mathcal{P}, \alpha \in Act_\tau . Q \xrightarrow{\alpha} Q' \implies \exists P' \in \mathcal{P} . P \xrightarrow{\alpha} P' \wedge (Q', P') \in R$$

The union of all bisimulations is, in turn, a bisimulation. Given two processes P and Q we say that P is *bisimilar* to Q, $P \sim Q$, if there is a bisimulation R such that $(P, Q) \in R$. Moreover, if $P \sim Q$ then we say that P can simulate Q and viceversa, since the relation is symmetric. The *bisimilarity* relation $\sim$ is the largest bisimulation relation over a given transition system. Notice that the symbol $\sim$ has been used to relate random variables and distributions however, since in this chapter we do not consider timing aspects and consequently distributions, it is always to be meant as the bisimulation symbol.

Bisimulation is important for comparing processes, we discuss here a very simple CCS example of use of such an equivalence. Let us consider the two processes

$$P_1 \stackrel{def}{=} \alpha.\beta.P + \alpha.\gamma.Q \qquad\qquad P_2 \stackrel{def}{=} \alpha.(\beta.P \mid \gamma.Q)\,.$$

It is fairly easy to notice that P_1 can perform action α, then can perform either action β or γ both leading to P_1 again. Differently, P_2 has to choose between two identical non-deterministic choices how to perform action α and, once the next action is to be performed, its is determined since it is β or γ depending on which of the branch has been chosen to perform α. Of course, the two processes are not bisimilar since P_1 can be simulated by P_2 but not vice-versa. In FIGURE 8.1 the LTSs of the two processes are represented.

For an algebraic treatment of bisimulation equivalence and to reason in a compositional way, a bisimulation is required to be a *congruence*. To formalize that, let us recall the standard notion of context as defined by this abstract syntax.

Definition (CCS Contexts) *Contexts* in CCS are defined by the following grammar:

$$C \;:=\; \Box \;\mid\; \alpha.C \;\mid\; C + P \;\mid\; P + C \;\mid\; C \mid P \;\mid\; P \mid C$$

where $\alpha \in Act$, $P \in \mathcal{P}$ and $A \in \mathcal{N}$. We denote the set of all contexts by $\mathcal{C}$.

Contexts are defined similarly to processes with the addiction of the context $\Box$. The idea of a context C is to define processes up to the hole $\Box$ appearing in the context definition. We require a process to contain at most a single hole $\Box$. We can trivially define a notion of context application as follows

$$\Box[P] = P \qquad\qquad (\alpha.C)[P] = \alpha.C[P]$$
$$(C + P')[P] = C[P] + P' \qquad\qquad (P' + C)[P] = P' + C[P]$$
$$(C \mid P')[P] = C[P] \mid P' \qquad\qquad (P' \mid C)[P] = P' \mid C[P]$$

where $C[P]$ represents the substitution of the $\Box$ in C by means of P. Now, given this notion of context and the notion of bisimulation it is possible to prove the following theorem

Theorem 8.1.1. *Bisimulation is a congruence with respect to all* CCS *operations since*

$$\forall P, Q \in \mathcal{P}.\, P \sim Q. \implies \forall C \in \mathcal{C}.\, C[P] \sim C[Q]\,.$$

Proof. If we consider CCS without recursion the proof comes by observing that inference rules respect the De Simone format we recalled in Section 2.5. The extension of the proof to the case of recursion is standard, as discussed in (Aceto *et al.*, 2001). □

This result states that if two processes are equivalent, then in any possible context the processes are inserted, the compositions of the context with process are bisimilar. In other words, equivalence is robust with respect to context application.

8.1.1 An example CCS model

In this section we present a CCS model of a simple biological system. Let us consider an enzyme A and two molecules of type B and C; the enzyme catalyzes degradation of molecule B and molecules B and C can bind together producing a composite molecule $B{:}C$ which is immune to degradation catalyzed by A. Of course, this is a very raw model which would require in general many more molecules/enzymes subtitling more different dynamics, but for the sake of simplicity this model gives enough intuition about modeling biological systems with CCS.

If we write a reaction-based model of this very simple system we obtain the following two reactions

$$R_1 : A + B \overset{k_1}{\longmapsto} A \qquad\qquad R_2 : B + C \overset{k_2}{\longmapsto} B{:}C$$

and the corresponding deterministic ODE model

$$\frac{dA}{dt} = 0 \qquad\qquad \frac{dB}{dt} = -k_1 AB - k_2 BC$$
$$\frac{dC}{dt} = -k_1 BC \qquad\qquad \frac{dB{:}C}{dt} = k_2 BC\,.$$

Reaction R_1 models the degradation of a molecule B catalyzed by A, which is not consumed. Reaction R_2 models the creation of a complex $B{:}C$ as the composition of B and C. Even if in the CCS stochastic effects are not considered, we assume reactions to fire with kinetic constants k_1 and k_2, respectively. It is fairly easy to notice the correspondence between the reactions and the terms in the equations. If we write the stoichiometry matrix D we get the following $D \in \mathbf{N}^{4\times 2}$

$$D = \begin{bmatrix} \nu_1 & \nu_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -1 & -1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}.$$

In order to model such a system, we assume a process for all the possible molecules and enzyme involved, A, B, C and $B{:}C$. We denote the processes with the same letters as the molecules and enzymes, and we denote with α (resp. $\overline{\alpha}$) and β (resp. $\overline{\beta}$) the actions modeling R_1 and R_2, respectively.

The CCS processes are defined as

$$A \overset{def}{=} \alpha.A \qquad\qquad B \overset{def}{=} \overline{\alpha}.0 + \beta.B{:}C$$
$$C \overset{def}{=} \overline{\beta}.0 \qquad\qquad B{:}C \overset{def}{=} 0\,.$$

Notice that the definition of the process $B{:}C$ is 0 since it appears only as a product of the events we want to model. We show now some steps of derivations for the a whole system described by a single copy of A, B and C

$$A \mid B \mid C$$

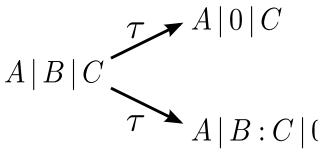

FIGURE 8.2: The LTS for the CCS 2-reactions model.

corresponding to the initial state vector $\mathbf{x}_0 = (1, 1, 1, 0)^T$.

We assume reaction R_1 to fire first. To this extent, the semantics permits to observe derivations as

$$A \xrightarrow{\alpha} A \qquad\qquad B \xrightarrow{\overline{\alpha}} 0$$

which compose as follows

$$A \mid B \mid C \xrightarrow{\tau} A \mid 0 \mid C\,.$$

Correctly, such configuration corresponds to the vector

$$\mathbf{x}_0 + \nu_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

which represents the fact that a molecule B has been degraded.

Differently, if R_2 fires first, then the similar derivations which can be observed can be summarized as

$$A \mid B \mid C \xrightarrow{\tau} A \mid B{:}C \mid 0$$

since $B \xrightarrow{\beta} B{:}C$ and $C \xrightarrow{\overline{\beta}} 0$. Correctly, such derivations correctly model the fact that

$$\mathbf{x}_0 + \nu_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

which represents the fact that a complex $B{:}C$ has been created by consuming a C. In FIGURE 8.2 a graphical representation of the LTS for this model is given. In such a figure are represented only the transitions which model the firing of a reaction, namely those with τ as a label.

8.2 CCS with delay-as-duration approach (CCSD)

In the following, we concentrate on actions modeling reactions following the delay-as-duration approach discussed in CHAPTER 4. We refer to this algebra as CCS *with delay-as-duration approach* (CCSD).

Processes of CCSD are described by the syntax of CCS processes. This is perfectly reasonable since delays are properties of the actions which can be performed by the processes of the algebra, so we can use the same CCS syntax.

8.2.1 Process configurations in CCSD

The use of non-instantaneous actions requires to distinguish a process in which actions are to be started from a process in which an action is started and has to complete. Moreover, the use of dyadic communication requires to couple together processes performing a started non-instantaneous action, so that when we compositionally derive transitions for the completion of the action, we can

ensure that we compose the derivations of the right processes. To this extent, *handshaking* is possible between any action α in P and its complementary action $\overline{\alpha}$ in Q. The handshaking has to be performed to assign to an instance of communication a unique identifier which may be used to compose derivations modeling the completion of the started action. Consequently, at any time of a computation, P can be in a configuration in which one of its actions is currently running. We model a process in which an action is started and has to be completed by introducing a notion of process configuration.

Definition (CCSD Process Configurations) *Process configurations* of CCSD are defined by the following grammar:

$$C_P \;:=\; [\alpha]^l.P \quad \mid \quad C_P \mid C_P \quad \mid \quad P$$

where $\alpha \in Act$, $l \in \mathbb{N}$ and $P \in \mathcal{P}$. We denote the set of all possible process configurations as $\mathcal{C}_{\mathcal{D}}$.

Any process $P \in \mathcal{P}$ is also in a valid configuration, hence $\mathcal{P} \subset \mathcal{C}$. However, a process configuration may contain actions denoted by a different prefix. In particular, the configuration $[\alpha]^l.P$ is the configuration reached by $\alpha.P$ after α has started. The new argument $l \in \mathbb{N}$ is a natural number that identifies the running action. Notice that these identifiers, which have to be unique, are computed by the handshaking performed before the start of an action and, once an action is to be completed, they are used to compose the derivations modeling such completion. By the definition of the semantics it will be clear how both the partners share the same identifier for the started action. Finally, notice that the configuration $C_P + C_P$ is not valid. The intuition for this is that, every time an action starts in summation, then the choice is resolved at that time since the action is non-instantaneous but it is performed exclusively by the process. This consideration is valid when considering delay-as-durations, since reactants are exclusively involved in the reaction and cannot have other interaction being consumed by the firing. More complex notions of delays (i.e. the purely delayed) may require more complex considerations.

Let us define by structural recursion an auxiliary function $Id : \mathcal{C}_{\mathcal{D}} \mapsto \wp(\mathbb{N})$ as follows:

$$\begin{aligned} Id([\alpha]^l.P) &= \{l\} \\ Id(C_P \mid C'_P) &= Id(C_P) \cup Id(C'_P) \\ Id(P) &= \emptyset\,. \end{aligned}$$

The value $Id(C_P)$ denotes the set of the identifiers of the actions in the configuration C_P that are running. For instance, given a configuration $C_P \equiv [\alpha]^l.P \mid \beta.Q \mid [\gamma]^{l'}.T$, the identifiers collected by function Id are given by $Id(C_P) = \{l, l'\}$.

In the next sections we define the semantics of CCSD by using the notions of CCS processes, CCSD process configurations and this auxiliary function.

8.2.2 A Structural Operational Semantics for CCSD

In this section we define a Structural Operational Semantics (SOS) for CCSD. All the inference rules we define are in TABLE 8.2 and in TABLE 8.3.

The main feature of the SOS we want is to support different relations for modeling non-instantaneous actions. We define the SOS in the Starting Terminating (ST) style (Bravetti & Gorrieri, 2002), as this permits to easily observe detached events as the start and the completion of an action. More precisely, we define a relation for modeling the start of an action and the coupling of processes; this is named as the *handshaking relation*. Also, we define a *completion relation* for modeling the completion of a started action.

The handshaking relation. This relation is used to model the start of an action and the coupling of the processes starting complementary actions. The handshaking relation is $\rightarrow_H \subseteq \mathcal{C}_{\mathcal{D}} \times \Theta^+ \times \mathcal{C}_{\mathcal{D}}$, where

$$\Theta^+ = \{(l, \alpha^+) \mid l \in \mathbb{N} \wedge \alpha \in Act_\tau\}$$

$$\texttt{(HPrefix)} \qquad \alpha.P \xrightarrow{(1,\alpha^+)}_H [\alpha]^1.P$$

$$\texttt{(HChoice)} \qquad \frac{P \xrightarrow{(l,\alpha^+)}_H P'}{P + Q \xrightarrow{(l,\alpha^+)}_H P'}$$

$$\texttt{(HConstant)} \qquad \frac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad A \stackrel{def}{=} P}{A \xrightarrow{(l,\alpha^+)}_H P'}$$

$$\texttt{(HCoop}_1\texttt{)} \qquad \frac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad l \notin Id(Q) \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{(l,\alpha^+)}_H P' \mid Q}$$

$$\texttt{(HCoop}_2\texttt{)} \qquad \frac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad l \in Id(Q) \qquad l' = min\{\mathbb{N} - Id(P \mid Q)\} \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{(l',\alpha^+)}_H P'[l'/l] \mid Q}$$

$$\texttt{(HCoop}_3\texttt{)} \qquad \frac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad Q \xrightarrow{(l',\overline{\alpha}^+)}_H Q' \qquad l'' = min\{\mathbb{N} - Id(P \mid Q)\}}{P \mid Q \xrightarrow{(l'',\tau^+)}_H P'[l''/l] \mid Q'[l''/l']}$$

TABLE 8.2: The handshaking relation $\rightarrow_H \subseteq \mathcal{C}_\mathcal{D} \times \Theta^+ \times \mathcal{C}_\mathcal{D}$ for CCSD.

and l represents the identifier assigned to the started action α and the use of the superscript "+" comes from the definition of the semantics in the ST-style to denote the start of an action. The SOS rules in TABLE 8.2 are at the basis of the definition of $\rightarrow_H$. We implicitly assume the rules symmetric with respect to (HChoice), ($\texttt{HCoop}_1$) and ($\texttt{HCoop}_2$). Unless explicitly specified, actions range over Act.

Rule (HPrefix) models the start of an action α. At any time a process with prefix α can start action α moving to a configuration in which it cannot perform the same action anymore, i.e. the configuration $[\alpha]^1.P$. Such a configuration, together with the one describing the process performing the complementary action, has to be uniquely identified by a natural number representing the identifier of the just started action. At this step, the process simply chooses 1 as unique identifier. All our choices for assigning identifiers to actions are inspired to those of (Bravetti & Gorrieri, 1999), which ensure that the portion of LTS rooted in a given process is finite. The rules for binary operator $\mid$ solve conflicts of colliding identifiers, if any.

Rule (HChoice) combines the start of an action with operator $+$. As in classic CCS, the choice is resolved at the start of an action, hence $P + Q$ becomes P' if P becomes P'. The value for l is used as a label of the transition.

Rule (HConstant) is the standard recursion rule for the handshaking relation. Rules ($\texttt{HCoop}_1$), ($\texttt{HCoop}_2$) and ($\texttt{HCoop}_3$) combine the start of an action with the operator $\mid$. Rules ($\texttt{HCoop}_1$) and ($\texttt{HCoop}_2$) model an autonomous move by one of the two processes, and deal with identifiers as follows. In rule ($\texttt{HCoop}_1$) there is no conflict (actions started in P do not share any identiÞer with actions started in Q), hence nothing has to be done. Differently, rule ($\texttt{HCoop}_2$) resolves conflicts of shared identifiers. The policy by which we choose the new fresh identifier, along the line of (Bravetti & Gorrieri, 1999), is such that the resulting LTS is finite. More precisely, the use of the set $\mathbb{N} - Id(P \mid Q)$ is such that we consider, in the process of renaming an identifier in P', the only set of identifiers not used in $P \mid Q$ and, from that, the choice of extracting the minimum value is such that the LTS is finite branching. We use this strategy in all the rules where we have to resolve some conflicts. By $P'[l'/l]$ we denote the classical renaming of all the occurrences of l by means of l'. We assume the classical distributivity of renaming over binary operators. Finally, as in classical process algebras, we do not force P and Q to handshake, since P could handshake with a further process composed in parallel with $P \mid Q$. Notice that here we may have a conflict

$$\texttt{(CPrefix)} \quad [\alpha]^l.P \xrightarrow{(l,\alpha^-)}_C P$$

$$\texttt{(CCoop}_1\texttt{)} \quad \frac{P \xrightarrow{(l,\alpha^-)}_C P' \qquad l \notin Id(Q) \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{(l,\alpha^-)}_C P' \mid Q}$$

$$\texttt{(CCoop}_2\texttt{)} \quad \frac{P \xrightarrow{(l,\alpha^-)}_C P' \qquad Q \xrightarrow{(l,\overline{\alpha}^-)}_C Q'}{P \mid Q \xrightarrow{(l,\tau^-)}_C P' \mid Q'}$$

TABLE 8.3: The completion relation $\rightarrow_C \subseteq \mathcal{C}_\mathcal{D} \times \Theta^- \times \mathcal{C}_\mathcal{D}$ for CCSD.

even if in Q the action associated with the colliding identifier is the complementary action $\overline{\alpha}$. Rule $\texttt{(HCoop}_3\texttt{)}$ models the handshaking by assigning to this particular instance of synchronization a new fresh identifier l'' chosen with the same policy used to resolve conflicts in the previous rules. The renaming of both old identifiers with the newly generated is due to the fact that, in general, the two processes have two different candidate identifiers, namely l and l'. The system in this case exhibits the internal action τ. By applying this rule, the two processes terminated this *handshaking phase*.

The completion relation. This relation is used to model the completion of a started action. The completion relation is $\rightarrow_C \subseteq \mathcal{C}_\mathcal{D} \times \Theta^- \times \mathcal{C}_\mathcal{D}$, where

$$\Theta^- = \{(l, \alpha^-) \mid l \in \mathbb{N} \wedge \alpha \in Act_\tau\}$$

and l represents the identifier that was assigned to the completed action α when it was started, and the use of the superscript "-" comes from the definition of the semantics in the ST-style. The rules presented in TABLE 8.3 are at the basis of the definition of $\rightarrow_C$. We implicitly assume a rule symmetric to $\texttt{(CCoop}_1\texttt{)}$.

Rule (CPrefix) describes the completion of an action. When it completes, the process is substituted by its continuation P. In the label, the identifier l is needed to couple this process with the one performing the corresponding complementary action $\overline{\alpha}$, which has the same identifier l because of the handshaking.

Rule $\texttt{(CCoop}_1\texttt{)}$ states that the completion of an action in P affects a parallel composition $P \mid Q$ in the classical CCS fashion, namely propagating the completion as a derivation of $P \mid Q$. Notice that, since it is required that once two processes have performed handshaking they derive composition of completion, then we require $l \notin Id(Q)$. Notice that this is perfectly equivalent of stating that a derivation as $Q \xrightarrow{(l,\overline{\alpha}^-)}_C Q'$ can not be performed by Q. We remark that, accordingly to the De Simone format of SOS rules we recalled in SECTION 2.5, negative premises are not allowed. Rule $\texttt{(CCoop}_2\texttt{)}$ models the case in which both P and Q complete actions that were coupled. As in classical process algebras, the whole system $P \mid Q$ exhibits an internal action τ.

Notice that there is no rule for a configuration defined as a constant $A \stackrel{def}{=} P$, this because a process associated to a constant is, by definition, not a process configuration, so it contains no running actions and consequently it can not perform any completion derivation.

Finally, once that we defined the two relations, we can define the semantics of CCSd.

Definition (CCSD Semantics) The *semantics* of CCSD is given by the LTS $(\mathcal{C}_\mathcal{D}, \Theta^+ \cup \Theta^-, \rightarrow_H \cup \rightarrow_C)$ where $\rightarrow_H$ and $\rightarrow_C$ are the minimal relations satisfying the rules given in TABLE 8.2 and Table 8.3, respectively.

On timing aspects in CCS with non-instantaneous actions In the semantics we defined, it is possible that two processes P and Q start a synchronization on α and $\overline{\alpha}$ once assigned an

identifier l, subsequently two copies of the same processes start a synchronization with same actions and are assigned an identifier l'. In our context of application actions model reactions, and hence reactions started first should complete first. It is not the case in our LTS since transitions of the completion relation for l'' can be derived independently of l'. This seems quite controversial at first glimpse, but in the end it is correct once we give a proper interpretation of a non-instantaneous action in CCSd. In order to state this, we need to discuss some considerations on timing aspects in this semantics.

As recalled in Section 2.1 the minimum of two exponentially distributed random variables is an exponential random variable with parameter equal to the summation of the parameters of the single variables. Moreover, in both the SSA and the DSSAs we defined exponential times of firing of reactions with parameters given by the summation of the propensity functions evaluated under proper conditions. In this sense, in the algebraic definitions of the equations ruling the probability distributions for the values of τ we can use the property of minimum of exponential distributions. This is what is done in the *First Reaction Method* (Gillespie, 1977), an algorithm *equivalent* to the SSA. In such an algorithm, to each reaction is a associated a putative time for the next reactions, and then the algorithm takes the minimum among all the times, and fires the associated reaction. More formally, given a system with M reactions and $i-$th propensity function evaluating to λ_i, the set of times

$$\{\tau_i \sim \mathcal{E}xp\,(\lambda_i) \mid i = 1, \ldots, M\}$$

is defined and the next reaction to fire is R_j if

$$j = \min\{\tau_i \sim \mathcal{E}xp\,(\lambda_i) \mid i = 1, \ldots, M\}\,.$$

Clearly, choosing a unique $\tau \sim \mathcal{E}xp\,(\sum_{i=1}^{n} \lambda_i)$ is equivalent, and is what is done by the SSA we introduced. This gives us the opportunity to think about firing times *local* to rules, instead of a firing time *global* to the system.

In this sense, we can associate times to each of all the reactions in the system. In CCS, this means that once a reaction is enabled, we can think of it consuming its exponentially distributed stochastic time. Once a reaction fires the system moves, the memoryless property is such that times are not affected by the fact that a reaction fired. Hence there is no need to store any residual time, and in fact in the stochastic semantics we do not consider explicit time. Moreover, in terms of stochastic process algebra the combination of the memoryless property and the notion of local time are necessary conditions preserving the well-known *expansion law* which underlies the interleaving semantics.

When moving to a framework with delays things do not get easier since it is fairly intuitive to notice the importance of storing residual times, as it is for instance in the definition of the non-Markovian processes we recalled in Section 2.2. To this extent, it would have been possible to define a semantics with an explicit notion of time, for instance by using clocks, but that is not what we did. In fact, we defined a semantics for CCSd which is without explicit time. In the semantics when an action starts, it must be seen as having generic duration built as composition of stochastic and deterministic times. The result is that this gives rise to situations in which the ordering of start of reactions does not imply any ordering for their completion, which is what actually is observable in our LTS. In fact, if with duration we had meant only the deterministic time than our LTS would have contained incorrect transitions. Also, composition with parallel operator would not have been straightforward.

Another advantage of this interpretation of durations is that classical results on bisimulation, which hold in CCS, still hold in CCS with non-instantaneous actions, as we discuss in the next sections.

8.2.3 Bisimulation in CCSd

In this section, we prove an important result on bisimulation for CCSd processes. We start by rephrasing the definition of bisimulation on process configurations of CCSd.

Definition (CCSD Bisimulation) Given a LTS $(\mathcal{C}_{\mathcal{D}}, \Theta^+ \cup \Theta^-, \rightarrow_H \cup \rightarrow_C)$ a binary relation $R \subseteq \mathcal{C}_{\mathcal{D}} \times \mathcal{C}_{\mathcal{D}}$ is a *bisimulation* if, for any $(C_1, C_2) \in R$, the following propositions hold:

$$\forall C_1' \in \mathcal{C}_{\mathcal{D}}, \ell \in \Theta^+ \cup \Theta^-. C_1 \xrightarrow{\ell}_r C_1' \wedge r \in \{H, C\} \implies \exists C_2' \in \mathcal{C}_{\mathcal{D}}. C_2 \xrightarrow{\ell}_r C_2' \wedge (C_1', C_2') \in R$$
$$\forall C_2' \in \mathcal{C}_{\mathcal{D}}, \ell \in \Theta^+ \cup \Theta^-. C_2 \xrightarrow{\ell}_r C_2' \wedge r \in \{H, C\} \implies \exists C_1' \in \mathcal{C}_{\mathcal{D}}. C_1 \xrightarrow{\ell}_r C_1' \wedge (C_2', C_1') \in R$$

Again, the union of all bisimulations is, in turn, a bisimulation and we extend all the considerations we did on CCS bisimulation to CCSD bisimulation. As we did for CCS, we investigate whether bisimulation is a congruence in CCSD. To this extent, we can notice that the definition of CCS context is valid for CCSD since the two algebras use the same syntax of processes and the syntax of contexts is related to processes, not to process configurations. As a consequence, we can prove the following theorem.

THEOREM 8.2.1. *Bisimulation is a congruence with respect to all* CCSD *operations since*

$$\forall P, Q \in \mathcal{C}_{\mathcal{D}}. P \sim Q. \implies \forall C \in \mathcal{C}. C[P] \sim C[Q]\,.$$

Proof. This proof is analogous to the proof of theorem for CCS since CCSD inference rules respect the De Simone format we recalled in SECTION 2.5. Again, the extension of the proof to the case of recursion is standard, as discussed in (Aceto *et al.*, 2001). □

Such result in the context of CCSD means that adding delays in this way is a robust operation with respect to classical bisimulation equivalence.

8.2.4 An example CCSD model

In this section we rephrase the simple CCS model we introduced in the previous section in the context of CCSD. We assign to reaction R_1 a delay σ and to reaction R_2 a delay σ'

$$R_1 : A + B \xmapsto{k,\sigma} A \qquad\qquad R_2 : B + C \xmapsto{k',\sigma'} B{:}C$$

so that the deterministic corresponding model is given by the following DDEs

$$\frac{dA}{dt} = 0 \qquad\qquad \frac{dB}{dt} = -kA(t-\sigma)B(t-\sigma) - k'B(t-\sigma')C(t-\sigma')$$
$$\frac{dC}{dt} = -k'B(t-\sigma')C(t-\sigma') \qquad\qquad \frac{dB{:}C}{dt} = k'B(t-\sigma')C(t-\sigma')\,.$$

In this case, the algebraic representation of the reactions is as follows

$$\nu_1 = \nu_1^r + \nu_1^p = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$
$$\nu_2 = \nu_2^r + \nu_2^p = \begin{pmatrix} 0 \\ -1 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

We can use the same CCS processes specification, the initial state $A \mid B \mid C$ and we can show the derivations corresponding to the events modeled by the corresponding CCS derivations. If reaction R_1 fires first, then the semantics permits to observe handshaking derivations as

$$A \xrightarrow{(1,\alpha^+)}_H [\alpha]^1 .A \qquad\qquad B \xrightarrow{(1,\overline{\alpha}^+)}_H [\overline{\alpha}]^1 .0$$

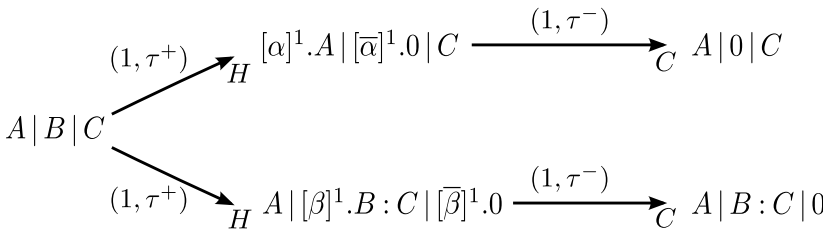

FIGURE 8.3: The LTS for the CCSD 2-reactions model.

which compose as follows

$$A \mid B \mid C \xrightarrow{(1,\tau^+)}_H [\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 \mid C \,.$$

Notice that, once R_1 is started, in the DDA we know that molecules A and B are locked until the reaction completes and, in CCSD, this is what happens since configuration $[\alpha]^1 .A \mid [\overline{\alpha}]^1 .0$ can only derive transitions of the completion relation. Such a configuration corresponds to the vector

$$\mathbf{x}_0 + \nu_1^r = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} .$$

Indeed, the next possible derivations are such that

$$[\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 \mid C \xrightarrow{(1,\tau^-)}_C A \mid 0 \mid C$$

which correctly models the fact that

$$\mathbf{x}_0 + \nu_1^r + \nu_1^p = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} .$$

Differently, if R_2 fires first similar derivations can be observed

$$A \mid B \mid C \xrightarrow{(1,\tau^+)}_H A \mid [\beta]^1 .B{:}C \mid \overline{[\beta]}^1 .0 \xrightarrow{(1,\tau^-)}_C A \mid B{:}C \mid 0$$

which correctly model the fact that

$$\mathbf{x}_0 + \nu_2^r + \nu_2^p = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} .$$

In FIGURE 8.3 a representation of the LTS for this system is given. As for the example of CCS, in such a figure are represented only the transitions which model the firing of a reaction, namely those with τ as a label.

8.3 CCS with purely delayed approach (CCSP)

In the following, we concentrate on actions modeling reactions following a purely delayed approach in its most precise definition discussed in CHAPTER 7. We refer to this algebra as CCS *with purely delayed approach* (CCSP).

8.3.1 Process configurations in CCSp

As for CCSd, we assume the same syntax of processes of CCS. However, when moving from CCSd to CCSp we need to refine the notion of process configurations. Indeed, the main feature of the purely delayed approach is that reactants, even if involved in some reaction already started, can take part in other reactions, which could start before the completion of the running ones. This means that, from the point of view of the CCS paradigm, a process P may start an action α and, before completing such an action, another action β may be started by P. Sometimes, both α and β belong to the same summation in P. This means that we must support configurations where two or more actions are started, and not completed, within the same summation. These actions are said to *compete for completion*. Moreover, along the line of the mPDa, the first competing action to complete interrupts all the other actions and co-actions, unlocking the involved partners. Practically, from the point of view of the configurations, this requires to extend the CCSd notion of configuration to the summation operator. Indeed, we define CCSp process configurations as follows.

Definition (CCSp Process Configurations) Process configurations of CCSp are defined by the following grammar:

$$C_P \;:=\; [\alpha]^l.P \;\;\mid\;\; C_P + C_P \;\;\mid\;\; C_P \mid C_P \;\;\mid\;\; P$$

where $\alpha \in Act$, $l \in \mathbb{N}$ and $P \in \mathcal{P}$. We denote the set of all possible process configurations as $\mathcal{C}_\mathcal{P}$.

The meaning of the configurations is the same of CCSd for all but the configuration $C_P + C_P$. Such a configuration, which is not valid in CCSd, models the fact that in CCSp actions can compete for completion inside a process, as depicted by the summation operator. We assume the function Id defined for CCSd to be trivially extended to this abstract syntax by adding the case definition $Id(C_P + C'_P) = Id(C_P) \cup Id(C'_P)$, hence $Id : \mathcal{C}_\mathcal{P} \mapsto \wp(\mathbb{N})$. By using this function an action α in $\alpha.P + C_P$ is said to compete with all the actions in $Id(C_P)$.

In the context of CCSp, we also assume a function used to *interrupt* a currently running set of actions in a process. Such a function is $Unlock : \mathcal{C}_\mathcal{P} \times \wp(\mathbb{N}) \mapsto \mathcal{C}_\mathcal{P}$ and is defined by structural recursion as follows

$$\begin{aligned}
Unlock([\alpha]^l.P, L) &= \begin{cases} \alpha.P & \text{if } l \in L \\ [\alpha]^l.P & \text{otherwise} \end{cases} \\
Unlock(C_P \mid C'_P, L) &= Unlock(C_P, L) \mid Unlock(C'_P, L) \\
Unlock(C_P + C'_P, L) &= Unlock(C_P, L) + Unlock(C'_P, L) \\
Unlock(P, L) &= P\,.
\end{aligned}$$

Intuitively, such a function interrupts all the actions whose identifiers are contained in its input set L, and leaves unmodified the others. Practically, this means that the process backtracks to a configuration in which such actions are not started.

We state some intuitive properties of the $Unlock$ function.

Proposition 8.3.1. *Let $L \in \wp(\mathbb{N})$, the following equations hold*

$$\begin{aligned}
L \supseteq Id(C_P) &\implies Unlock(C_P, L) = Unlock(C_P, Id(C_P)) \\
L \subseteq Id(C_P) &\implies Unlock(C_P, L) = Unlock(C_P, Id(C_P) \cap L) \\
Unlock(C_P, Id(C_P) \cap L) &= Unlock(C_P, L)\,.
\end{aligned}$$

Obviously, $Unlock(C_P, Id(C_P)) \in \mathcal{P}$ since it denotes a process in which no actions are running. Differently, in the second equation $Unlock(C_P, L)$ may contain running actions. The third equation is a consequence of the combination of the first two. So for instance, given a configuration $C_P \equiv$

(HPrefix) $\alpha.P \xrightarrow{(1,\alpha^+)}_H [\alpha]^1.P$

(HChoice$_1$) $\dfrac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad l \notin Id(Q)}{P+Q \xrightarrow{(l,\alpha^+)}_H P'+Q}$

(HChoice$_2$) $\dfrac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad l \in Id(Q) \qquad l' = min\{\mathbb{N} - Id(P+Q)\}}{P+Q \xrightarrow{(l',\alpha^+)}_H P'[l'/l]+Q}$

(HConstant) $\dfrac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad A \stackrel{def}{=} P}{A \xrightarrow{(l,\alpha^+)}_H P'}$

(HCoop$_1$) $\dfrac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad l \notin Id(Q) \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{(l,\alpha^+)}_H P' \mid Q}$

(HCoop$_2$) $\dfrac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad l \in Id(Q) \qquad l' = min\{\mathbb{N} - Id(P \mid Q)\} \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{(l',\alpha^+)}_H P'[l'/l] \mid Q}$

(HCoop$_3$) $\dfrac{P \xrightarrow{(l,\alpha^+)}_H P' \qquad Q \xrightarrow{(l',\overline{\alpha}^+)}_H Q' \qquad l'' = min\{\mathbb{N} - Id(P \mid Q)\}}{P \mid Q \xrightarrow{(l'',\tau^+)}_H P'[l''/l] \mid Q'[l''/l']}$

TABLE 8.4: The handshaking relation $\rightarrow_H \subseteq \mathcal{C}_\mathcal{P} \times \Theta^+ \times \mathcal{C}_\mathcal{P}$ for CCSP.

$[\alpha]^l.P \mid \beta.Q \mid [\gamma]^{l'}.T \mid Q$, we have that

$$\begin{aligned} Unlock(C_P, \{l\}) &= \alpha.P \mid \beta.Q \mid [\gamma]^{l'}.T \mid Q \\ Unlock(C_P, \{l'\}) &= [\alpha]^l.P \mid \beta.Q \mid \gamma.T \mid Q \\ Unlock(C_P, Id(C_P)) &= \alpha.P \mid \beta.Q \mid \gamma.T \mid Q \\ Unlock(C_P, \emptyset) &= C_P\,. \end{aligned}$$

Notice that this function is not needed in CCSD because there is no notion of competing actions, and hence no need of interrupt running actions.

In the next sections we define the semantics of CCSP by using the notions of CCS process, CCSP process configuration and these auxiliary functions.

8.3.2 A Structural Operational Semantics for CCSP

In this section we define a Structural Operational Semantics for CCSP; all the rules are in TABLE 8.4 and in TABLE 8.5. We want this SOS to have the same features we required for the one of CCSD. Indeed, we again define the SOS in the ST-style (Bravetti & Gorrieri, 2002), and similarly we define an handshaking and a completion relation. Although this similarity, these relations in the context of CCSP are less intuitive than the ones in CCSD.

The handshaking relation. This relation is used to model the start of an action and the coupling of the processes starting complementary actions, as it was in CCSD. The handshaking relation is $\rightarrow_H \subseteq \mathcal{C}_\mathcal{P} \times \Theta^+ \times \mathcal{C}_\mathcal{P}$, where Θ^+ has the same meaning as the one used in the SOS of CCSD. The SOS rules in TABLE 8.4 are at the basis of the definition of$\rightarrow_H$. We implicitly assume the rules symmetric with respect to (HChoice$_1$), (HChoice$_2$), (HCoop$_1$) and(HCoop$_2$).

Most of these rules are the same appearing in the handshaking relation for CCSD but the new inference rules (HChoice$_1$) and (HChoice$_2$); they combine the start of an action with operator +,

$$\texttt{(CPrefix)} \quad [\alpha]^l.P \xrightarrow{(l,\alpha^-,\emptyset)}_C P$$

$$\texttt{(CChoice)} \quad \frac{P \xrightarrow{(l,\alpha^-,L)}_C P'}{P+Q \xrightarrow{(l,\alpha^-,L\cup Id(Q))}_C P'}$$

$$\texttt{(CCoop}_1\texttt{)} \quad \frac{P \xrightarrow{(l,\alpha^-,L)}_C P' \qquad l \notin Id(Q) \qquad \alpha \in Act_\tau}{P \mid Q \xrightarrow{(l,\alpha^-,L\setminus Id(Q))}_C P' \mid Unlock(Q,L)}$$

$$\texttt{(CCoop}_2\texttt{)} \quad \frac{P \xrightarrow{(l,\alpha^-,L)}_C P' \quad Q \xrightarrow{(l,\overline{\alpha}^-,M)}_C Q' \quad N = L \cap M}{P \mid Q \xrightarrow{(l,\tau^-,(L\cup M)\setminus N)}_C Unlock(P',M) \mid Unlock(Q',L)}$$

TABLE 8.5: The completion relation $\rightarrow_C \subseteq \mathcal{C}_\mathcal{P} \times \Theta_C^- \times \mathcal{C}_\mathcal{P}$ for CCSP.

as required by the augmented definition of process configurations. In rule $\texttt{(HChoice}_2\texttt{)}$ the identifier l of the action α started by process P has no conflicts with the identifiers of the competing actions running in process Q. Differently, in the case of rule $\texttt{(HChoice}_2\texttt{)}$ a conflict does exist, which implies that a fresh identifier l' replaces l. We use the same strategy used in the definition of the SOS for CCSD. An important consideration regarding both the rules is worth discussing. The rule defines the start of an action in a process summation but, differently from CCSD or classical process algebras, the summation is not resolved as a choice at this moment. The capability of having competing actions is at the basis for the choices we made in the definition of CCSP. A process $P \stackrel{def}{=} P_1 + \ldots + P_n$ can start multiple actions in parallel, but can be involved in each action at most once at a time, as in the MPDA. As we said, in classical process algebras, this is not possible since an action, when starts, determines the future process in which P is transformed. In this sense, the summation operator of CCSP is not a classical choice for the reason that the competing actions compete for their completion, and, then, the semantics of the completion, and hence the semantics of the CCSP summation, depends on the action which first completes.

The completion relation. This relation is used to model the completion of an action. Although this relation has the same aim as the one defined for CCSP, its definition is more complex because it has to potentially interrupt competing actions. The completion relation is $\rightarrow_C \subseteq \mathcal{C}_\mathcal{P} \times \Theta_C^- \times \mathcal{C}_\mathcal{P}$, where

$$\Theta_C^- = \{(l,\alpha^-,N) \mid l \in \mathbb{N} \wedge \alpha \in Act_\tau \wedge N \in \wp(\mathbb{N})\}$$

and l and α have the usual meaning of Θ^- in the SOS of CCSD, and N is the set of the identifiers of the competing actions that are interrupted by the termination of α. The rules presented in TABLE 8.5 are at the basis of the definition of $\rightarrow_C$. We implicitly assume rules symmetric to $\texttt{(CChoice)}$ and $\texttt{(CCoop}_1\texttt{)}$.

Rule $\texttt{(CPrefix)}$ describes the completion of an action; the process is substituted by its continuation P. In the label, the identifier l is needed to couple this process with the one performing the corresponding complementary action $\overline{\alpha}$, which has the same identifier l because of the handshaking, and $\emptyset$ states that no action is interrupted.

Rule $\texttt{(CChoice)}$ states that the completion of an action in P affects a summation $P+Q$ so that all actions running in Q should be interrupted. This is obtained by adding to the set of labels of actions interrupted L, the set of actions currently running in the process Q which disappears by the completion of the action in P, hence the exhibited set of labels becomes $L \cup Id(Q)$. Notice that, at this time, the choice is resolved as in classical process algebras.

Rule $\texttt{(CCoop}_1\texttt{)}$ states that the completion of an action in P affects a parallel composition $P \mid Q$ so that all actions running in Q that are coupled with actions interrupted in P, $Id(Q) \cap L$, must

be interrupted as well. In this sense, since $Unlock(Q, Id(Q) \cap L) = Unlock(Q, L) = Q'$ we use as input of $Unlock$ directly L. The remaining set of actions to interrupt is hence $L \setminus Id(Q)$.

Rule $(\texttt{CCoop}_2)$ models the case in which both P and Q complete actions that were coupled. As in classical process algebras, the whole system $P \mid Q$ exhibits an internal action τ. Some of the actions required to be interrupted outside P, may be also required to be interrupted by Q. Such a set is denoted by N and can be removed from the set of actions that can be interrupted outside $P \mid Q$. The remaining set of actions, which have to be still interrupted by further composition with the parallel operator outside $P \mid Q$, is the set of those belonging to P and not to Q, and viceversa. Notice that actions required to be interrupted from Q in P, M, are effectively interrupted by $Unlock(P', M)$, and those required to be interrupted from P in Q, L, are effectively interrupted by $Unlock(Q', L)$.

Finally, once that we defined the two relations, we can define the semantics of CCSP.

Definition (CCSP Semantics) The *semantics* of CCSP is given by the LTS $(\mathcal{C}_{\mathcal{P}}, \Theta^{+} \cup \Theta^{-}_{C}, \rightarrow_{H} \cup \rightarrow_{C})$ where $\rightarrow_{H}$ and $\rightarrow_{C}$ are the minimal relations satisfying the rules given in TABLE 8.4 and TABLE 8.5, respectively.

We remark that the considerations we did about the assumption on the durations and the definition of the completion relation for CCSD still hold for this completion relation and this semantics. Again, a different notion of duration would have required a more complex set of rules for this relation. Moreover, this semantics is much more complex than the one for CCSD as effectively refers to a more complex notion of delay. In this sense, having kept this reasonably simple endorses our choices.

Before discussing CCSP bisimulation, we discuss the interruption of processes in CCSP via an example.

On actions interruption. Let us consider a generic CCSP process configuration of the form

$$[\alpha]^{l}.P + \Sigma \mid Q \mid [\overline{\alpha}]^{l}.P' + \Sigma' \mid R$$

in which actions α and $\overline{\alpha}$ are started and have been coupled after the handshaking. In such a configuration, α and $\overline{\alpha}$ compete with actions labeled in $Id(\Sigma)$ and $Id(\Sigma')$, respectively. Moreover, let us assume that

$$l' \in Id(Q) \cap Id(\Sigma') \quad l'' \in Id(R) \cap Id(\Sigma) \quad l''' \in Id(R) \cap Id(\Sigma')$$

namely there is an action with identifier l' running in Q and Σ', an action with identifier l'' running in R and Σ and an action with identifier l''' running in R and Σ'.

Once action α completes we have that all the actions in Σ have to be interrupted, so at least l'', so we can derive the following transition of the completion relation

$$[\alpha]^{l}.P + \Sigma \xrightarrow{(l, \alpha^{-}, Id(\Sigma))}_{C} P$$

which means that, once we evaluate $Unlock(Q, Id(\Sigma)) = Q'$ we can derive

$$[\alpha]^{l}.P + \Sigma \mid Q \xrightarrow{(l, \alpha^{-}, \rho)}_{C} P \mid Q'$$

where $\rho = Id(\Sigma) \setminus Id(Q)$. We can notice that $l'' \notin Id(\Sigma) \cap Id(Q)$, hence $l'' \in \rho$ as expected is exhibited as a label. This is correct since such an action is running also in R, similarly action with identifier l' is still running in Q'. As we know, we expect the completion of $\overline{\alpha}$ to interrupt the latter in Q', and the composition of the termination to interrupt the former in R.

Once we consider the completion of action $\overline{\alpha}$, we can derive the following transitions

$$[\overline{\alpha}]^{l}.P' + \Sigma' \mid R \xrightarrow{(l, \overline{\alpha}^{-}, \rho')}_{C} P \mid R'$$

where $\rho' = Id(\Sigma') \setminus Id(R)$ and $Unlock(R, Id(\Sigma')) = R'$. We can observe that by construction $l''' \in Id(R)$ so $l''' \notin R'$, moreover it still holds that $l' \in Id(Q')$ and $l'' \in Id(R')$. These two actions are correctly interrupted once that we compose the completion of α and $\overline{\alpha}$, in fact we derive

$$[\alpha]^l .P + \Sigma \mid Q \mid [\overline{\alpha}]^l .P' + \Sigma' \mid R \xrightarrow{(\tau, l^-, \rho'')}_C P \mid Q'' \mid P' \mid R''$$

where $Unlock(Q', \rho') = Q''$, $Unlock(R', \rho) = R''$ and $\rho'' = (\rho \cup \rho') \setminus (\rho \cap \rho')$. As expected, it is easy to verify that $l' \in \rho'$, $l'' \in \rho$ and hence $l' \notin Id(Q'')$ and $l'' \notin Id(R'')$. Finally, we can notice that

$$\begin{aligned} Q'' &= Unlock(Q', \rho') \\ &= Unlock(Unlock(Q, Id(\Sigma)), \rho') \\ &= Unlock(Unlock(Q, Id(\Sigma)), Id(\Sigma') \setminus (Id(\Sigma') \cap Id(R))) \\ &= Unlock(Unlock(Q, Id(\Sigma)), Id(\Sigma')) \\ &= Unlock(Q, Id(\Sigma) \cup Id(\Sigma')) \end{aligned}$$

where the last equations are valid since an identifier belongs exactly to two sets, so the identifiers in ρ' which shared between Q and Σ' are independent on those shared from Σ' and R. With similar arguments it is possible to verify that

$$R'' = Unlock(R, Id(\Sigma) \cup Id(\Sigma'))$$

so, by this considerations, we rewrote the starting process configuration in the new configuration

$$P \mid Unlock(Q, Id(\Sigma) \cup Id(\Sigma')) \mid P' \mid Unlock(R, Id(\Sigma) \cup Id(\Sigma')) .$$

Notice that if we had defined a *reduction* semantics instead of a SOS for CCSp, this transition would have completely defined the the completion relation for such a semantics.

8.3.3 Bisimulation in CCSp

In this section, we prove an important result on bisimulation for CCSp processes, analogous to the one we proved for CCSd. We start by rephrasing the definition of bisimulation on process configurations of CCSp.

Definition (CCSp Bisimulation) Given a LTS $(\mathcal{C}_\mathcal{P}, \Theta^+ \cup \Theta^-, \rightarrow_H \cup \rightarrow_C)$ a binary relation $R \subseteq \mathcal{C}_\mathcal{P} \times \mathcal{C}_\mathcal{P}$ is a *bisimulation* if, for any $(C_1, C_2) \in R$, the following propositions hold:

$$\forall C_1' \in \mathcal{C}_\mathcal{P}, \ell \in \Theta^+ \cup \Theta^- . C_1 \xrightarrow{\ell}_r C_1' \wedge r \in \{H, C\} \implies \exists C_2' \in \mathcal{C}_\mathcal{P} . C_2 \xrightarrow{\ell}_r C_2' \wedge (C_1', C_2') \in R$$

$$\forall C_2' \in \mathcal{C}_\mathcal{P}, \ell \in \Theta^+ \cup \Theta^- . C_2 \xrightarrow{\ell}_r C_2' \wedge r \in \{H, C\} \implies \exists C_1' \in \mathcal{C}_\mathcal{P} . C_1 \xrightarrow{\ell}_r C_1' \wedge (C_2', C_1') \in R$$

As expected, because of the similarity in the definition of the LTS for CCSp and CCSd, the bisimulation turns out to be similar to the one defined in CCSd. We extend all the considerations we did on CCS and CCSd bisimulations to CCSp bisimulation.

As we did for CCSd, we investigate whether CCSp bisimulation is a congruence or not. For the same reasons we discussed in CCSd we can use the definition of CCS contexts in CCSp, however the proof of congruence can not be done by observing that the SOS format is De Simone. More precisely, rules $(\texttt{CCoop}_1)$ and $(\texttt{CCoop}_2)$ do not satisfy such a format because of the use of function *Unlock*. In this sense this requires us to directly prove the congruence property or to change such inference rules with equivalent ones in the De Simone format; we decide to go through the latter possibility.

(IPrefix$_1$)	$[\alpha]^l.P \xrightarrow{\{l\}}_I \alpha.P$
(IPrefix$_2$)	$[\alpha]^l.P \xrightarrow{\emptyset}_I [\alpha]^l.P$
(IPrefix$_3$)	$\alpha.P \xrightarrow{\emptyset}_I \alpha.P$
(IConstant)	$A \xrightarrow{\emptyset}_I P'$
(INil)	$0 \xrightarrow{\emptyset}_I 0$
(IChoice)	$\dfrac{P \xrightarrow{L}_I P' \qquad Q \xrightarrow{M}_I Q'}{P+Q \xrightarrow{L\cup M}_I P'+Q'}$
(ICoop)	$\dfrac{P \xrightarrow{L}_I P' \qquad Q \xrightarrow{M}_I Q'}{P \mid Q \xrightarrow{L\cup M}_I P' \mid Q'}$

TABLE 8.6: The interruption relation $\rightarrow_I \subseteq \mathcal{C}_\mathcal{P} \times \wp(\mathbb{N}) \times \mathcal{C}_\mathcal{P}$ for CCSP.

An interruption relation. In order to define new SOS rules instead of (CCoop$_1$) and (CCoop$_2$) we define a relation working as function *Unlock*. The new relation, named the *interruption* relation, is used to model the interruption of a set of actions currently running in a process, as it was for function *Unlock*. The interruption relation is $\rightarrow_I \subseteq \mathcal{C}_\mathcal{P} \times \wp(\mathbb{N}) \times \mathcal{C}_\mathcal{P}$, where a label $M \in \wp(\mathbb{N})$ contains the identifiers of the actions that have been interrupted. Notice that, differently from function *Unlock*, this relation exhibits a set of identifiers of actions interrupted whereas function *Unlock* assumed such a set as input. The rules presented in TABLE 8.6 are at the basis of the definition of $\rightarrow_I$.

Before commenting the rules, a consideration is worth discussing. Function *Unlock* was quite intuitive in its definition since it was assuming as input the set of labels of actions to interrupt. Given a configuration C, there exist infinite sets L so that $Unlock(Q, L)$ is defined. Among all these sets, there are all those in $\wp(Id(Q))$ which give different output configurations, differently all the sets built as $\{n \in |n \notin Id(Q)\}$ give the same output of the input set $\emptyset$. In the case of inference rules there is no notion of input or output as in functions, but we define transitions between configurations. In this sense, the interruption relation can not be defined by using any input set analogous to the one used in function *Unlock* and, consequently, it is to be defined so that from a starting configuration it is possible to observe multiple derivations. In particular, it is required to observe all of those related to the sets we defined. Moreover, we require the rules to be in the De Simone format.

With this intuition we can comment the rules defining such a relation. At any time, a process in configuration $[\alpha]^l.P$ may interrupt the action it is currently performing. In these cases, treated with rule (IPrefix$_1$), it moves to a configuration in which the interrupted action α may start again, namely to configuration $\alpha.P$, the identifier l of the interrupted action is exhibited as a label of this transition. This information can be used to interrupt also the partner of this action, as we know that there is a partner in the system which, after terminating the handshaking phase, has been coupled with the same label l.

To get all the possible derivations from a configuration not all the actions have to be interrupted, so the process in configuration $[\alpha]^l.P$ must be able also to non-deterministically decide whether to interrupt or not. This case is described by rule (IPrefix$_2$). In support of this intuition we give an example; let us assume a process configuration $(P + \Sigma) \mid (Q + \Sigma') \mid S \mid R$, where both P and Q successfully complete an action. The actions to be interrupted are those currently running in both Σ and Σ', namely those with identifiers denoted by $Id(\Sigma) \cup Id(\Sigma')$. Let us assume that some of the actions that have to be interrupted in Σ and Σ' were coupled with some actions in S. In this case, also these actions in S should be interrupted as well. Moreover, S may be involved

in other actions currently running and coupled with actions in R. Indeed, these actions must not be interrupted. This means that from S the correct derivation with the interruption relation, in general, do not exhibit as label $Id(S)$, indeed it exhibits a strict subset of $Id(S)$. This implies that S must be able to autonomously decide which actions to interrupt, and this can be done by properly combining derivations of the interruption relation. The composition of the relations of the whole semantics provides the correctness, namely the fact that all and only those to interrupt are actually interrupted.

Also, a process which is not performing any action, namely a process in a configuration $\alpha.P$, does not interrupt any action, as stated by rule `(IPrefix`$_2$`)`. Similarly, rule `(IConstant)` models the fact that a process name can not stop any action, since it contains by definition no running actions.

Rules `(IChoice)` and `(ICoop)` simply collects the labels of the interrupted actions in a summation. Notice the interruption rule `(INil)`; it is quite unusual that a process 0 derives any transition however, for the considerations we discussed, we have to require that each process performs a derivation of this relation. To this extent, as we gave non-determinism to processes, and as we gave a unique choice for constants and processes $\alpha.P$, then we are required to give 0 the capability of deriving a transition which, effectively, does not change 0.

Now that we discussed the definition of the relation, we can prove its correspondence with function $Unlock$.

Theorem 8.3.2. $\forall C_P, C'_P \in \mathcal{C}, L \subseteq Id(C_P).\, Unlock(C_P, L) = C'_P \iff C_P \xrightarrow{L}_I C'_P$

Proof. We divide the proof by cases.

($\implies$) We prove by induction on the structure of C_P that

$$Unlock(C_P, L) = C'_P \implies C_P \xrightarrow{L}_I C'_P$$

- ($C_P \equiv [\alpha]^l.P$) If $L = \{l\}$ then $Unlock(C_P, \{l\}) = \alpha.P$ and we apply rule `(IPrefix`$_1$`)`. Similarly, if $L = \emptyset$ then $Unlock(C_P, \{l\}) = C_P$ and we apply rule `(IPrefix`$_2$`)` to C_P.
- ($C_P \equiv C_{P_1} + C_{P_2}$) Let L be a generic subset of $Id(C_P)$, in this case $Unlock(C_P, L) = Unlock(C_{P_1}, L) + Unlock(C_{P_2}, L)$, the inductive hypotheses are

  $$Unlock(C_{P_1}, L) = C'_{P_1} \implies C_{P_1} \xrightarrow{L}_I C'_{P_1}$$
  $$Unlock(C_{P_2}, L) = C'_{P_2} \implies C_{P_2} \xrightarrow{L}_I C'_{P_2}$$

  which permit to successfully apply SOS rule `(IChoice)`.
- ($C_P \equiv C_{P1} \mid C_{P_2}$) This case is analogous to the one for $+$.
- ($C_P \equiv P$) In this case, for any L we have $Unlock(C_P, L) = P$, so we can use derivations of the relation with proper combinations of all the rules but `(IPrefix`$_1$`)`.

($\impliedby$) We prove by induction on C_P that $C_P \xrightarrow{L}_I C'_P \implies Unlock(C_P, L) = C'_P$; since this case is almost equivalent to the previous one, we skip it.

□

Such a theorem states the equivalence between the relation and function $Unlock$, namely says that we can change the effect of the function by means of a proper derivation of the relation. In this sense, this gives us hints on how to proceed to prove that CCSp bisimulation is a congruence. If we assume the same definition of CCS contexts in CCSd, and this can be done since we are using the same syntax for both CCS and CCSd processes, it is possible to prove the following theorem.

Theorem 8.3.3. *Bisimulation is a congruence with respect to all* CCSp *operations since*

$$\forall P, Q \in \mathcal{C}_\mathcal{P}.\, P \sim Q. \implies \forall C \in \mathcal{C}_\mathcal{P}.\, C[P] \sim C[Q]\,.$$

$$\texttt{(CPrefix)} \quad [\alpha]^l.P \xrightarrow{(l,\alpha^-,\emptyset,\emptyset)}_C P$$

$$\texttt{(CChoice)} \quad \frac{P \xrightarrow{(l,\alpha^-,L,\emptyset)}_C P'}{P+Q \xrightarrow{(l,\alpha^-,L\cup Id(Q),\emptyset)}_C P'}$$

$$\texttt{(CCoop}_1\texttt{)} \quad \frac{\begin{array}{ccc} P \xrightarrow{(l,\alpha^-,L_1,L_2)}_C P' & Q \xrightarrow{M}_I Q' & l \notin Id(Q) \\ M \supseteq (Id(Q)\cap L_1) & V = M \setminus (L_1 \cap Id(Q)) & \alpha \in Act_\tau \end{array}}{P \mid Q \xrightarrow{(l,\alpha^-,L\setminus Id(Q),L_2\cup V)}_C P' \mid Q'}$$

$$\texttt{(CCoop}_2\texttt{)} \quad \frac{\begin{array}{ccc} P \xrightarrow{(l,\alpha^-,L_1,L_2)}_C P' & Q \xrightarrow{(l,\overline{\alpha}^-,M_1,M_2)}_C Q' & L_2 \subseteq M_1 \\ M_2 \subseteq L_1 & N_1 = L_1 \cap M_1 & N_2 = L_2 \cup M_2 \end{array}}{P \mid Q \xrightarrow{(l,\tau^-,((L_1\cup M_1)\setminus N_2)\setminus N_1,\emptyset)}_C P' \mid Q'}$$

TABLE 8.7: The new completion relation $\rightarrow_C \subseteq \mathcal{C}_\mathcal{P} \times \Theta^-_{C'} \times \mathcal{C}_\mathcal{P}$ for CCSP.

Proof. To prove this result we consider a new semantics of CCSP, equivalent to the old one, where the completion relation is slightly changed. We consider a new completion relation $\rightarrow_C \subseteq \mathcal{C}_\mathcal{P} \times \Theta^-_{C'} \times \mathcal{C}_\mathcal{P}$ defined with labels on an extended set

$$\Theta^-_{C'} = \{(l,\alpha^-,N_1,N_2) \mid l \in \mathbb{N} \wedge \alpha \in Act_\tau \wedge N_1,N_2 \in \wp(\mathbb{N})\}.$$

Set N_1 is the same of set N in Θ^-_C, set N_2 is a new set which describes the actions which have been interrupted on request of a process composed in parallel, as it is clear from the forthcoming explanations. The rules presented in TABLE 8.7 are at the basis of the definition of $\rightarrow_C$. As before, we implicitly assume rules symmetric to `(CChoice)` and $\texttt{(CCoop}_1\texttt{)}$.

We briefly comment these rules. Rule `(CPrefix)` and `(CChoice)` are not modified, the empty set as label is due to the fact that we are not considering parallel compositions in these rules. Other rules contain major differences with respect to the old ones. Rule $\texttt{(CCoop}_1\texttt{)}$ is such that it derives from Q a derivation of the interruption relation interrupting at least actions in $(Id(Q)\cap L_1)$, notice that this is quite different from the previous rule not only for the use of the relation instead of the function. In fact, in this case we interrupt a set of actions $M = V \cup (Id(Q) \cap L_1)$ which may contain more labels than those in L_1, if V is not empty. We recall that the original input of function *Unlock* was simply L_1. In this sense, the relation is used in a more powerful way so that we can define a new rule as $\texttt{(CCoop}_2\texttt{)}$. Such a rule says that in P actions interrupted are in L_1, and actions interrupted on request of the process composed in parallel, Q, are in L_2. To have a correct behavior we require to effectively have interrupted appropriate actions, namely $L_2 \subseteq M_1$. Obviously, M_1 and M_2 have the same meaning of L_1 and L_2. Actions to be interrupted on further compositions are those interrupted in both P and Q, once that those required by external compositions, N_2, and those interrupted by each other, N_1, have been removed. As expected, since $L_2 \subseteq M_1$ and $M_2 \subseteq L_1$, then the second label is empty.

We can now consider a LTS for CCSP where this relation is used, instead of the new one. Such a new LTS is $(\mathcal{C}_\mathcal{P}, \Theta^+ \cup \Theta^-_{C'}, \rightarrow_H \cup \rightarrow_C)$ where $\rightarrow_H$ and $\rightarrow_C$ are the minimal relations satisfying the rules given in TABLE 8.4 and TABLE 8.7, respectively. THEOREM 8.3.2 guarantees that the LTSs built by this new semantics is equivalent to the one built by the other, once we disregard this new label. As a consequence, the proof comes from noting that the new rules, including those of the interruption relation, are in the De Simone format. The proof comes by the same arguments of the proof for CCSD. □

Also in the case of CCSP, the meaning of such result is that it is possible to add delays respecting the purely delayed approach and preserving results valid in the non-delayed framework.

8.3.4 An example CCSp model

As we did for CCSd, we define the very same model of biological system we previously presented in the context of CCSp. We assume the same set of reactions, equations, processes and initial state for the CCSd model.

We show the corresponding CCSp derivation; we assume reaction R_1 to fire first. To this extent, the semantics permits to observe handshaking derivations as

$$A \xrightarrow{(1,\alpha^+)}_H [\alpha]^1 .A \qquad\qquad B \xrightarrow{(1,\overline{\alpha}^+)}_H [\overline{\alpha}]^1 .0 + \beta.B{:}C$$

which compose as follows

$$A \mid B \mid C \xrightarrow{(1,\tau^+)}_H [\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 + \beta.B{:}C \mid C\,.$$

Notice that, once R_1 is started, in the mPDA we know that molecule A, which is locked on reaction R_1, is not able to start any other reaction but, differently, molecule B is only locked on reaction R_1 and can take part in reaction R_2. As we said, in CCSp, this is expressed by the configuration $[\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 + \beta.B{:}C \mid C$. Indeed, such configuration corresponds to the vector

$$\begin{pmatrix} 1 \cdot \{R_1\} \\ 1 \cdot \{R_1\} \\ 1 \cdot \emptyset \\ 0 \end{pmatrix}.$$

where $1 \cdot R$ denotes that 1 molecule is involved in reactions in the set R. In this state, the process with markings can perform more choices: reaction R_1 can complete (leading to the same state obtained by completion of R_1 in the CCSd computation), or reaction R_2 can start. We discuss this second case, indeed we derive from the process the handshaking derivations

$$[\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 + \beta.B{:}C \mid C \xrightarrow{(2,\tau^+)}_H [\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 + [\beta]^2 .B{:}C \mid \left[\overline{\beta}\right]^2 .0$$

and this process correspond to the vector with markings

$$\begin{pmatrix} 1 \cdot \{R_1\} \\ 1 \cdot \{R_1, R_2\} \\ 1 \cdot \{R_2\} \\ 0 \end{pmatrix}.$$

In fact, here molecule B is involved (and consequently marked) in both the started reactions. From this process, it is possible to derive derivations for the completion relation either modeling the completion of R_1 (i.e $\alpha/\overline{\alpha}$) or R_2 (i.e. $\beta/\overline{\beta}$).

If we consider the first of the two possibilities, we expect to derive from the process the derivations

$$[\alpha]^1 .A \mid [\overline{\alpha}]^1 .0 + [\beta]^2 .B{:}C \mid \left[\overline{\beta}\right]^2 .0 \xrightarrow{(1,\tau^-,\emptyset)}_C A \mid 0 \mid C$$

built by unlocking C and B by means of the derivations

$$[\beta]^2 .B{:}C \xrightarrow{\{2\}}_I \beta.B{:}C \qquad\qquad \left[\overline{\beta}\right]^2 .0 \xrightarrow{\{2\}}_I C$$

or analogously by using the $Unlock$ function. As expected, this corresponds to the vector $(1, 0, 1, 0)^T$ where no markings are present. In FIGURE 8.4 a representation of the LTS for this system is given. As for the example of CCSd, in such a figure are represented only the transitions which model the firing of a reaction.

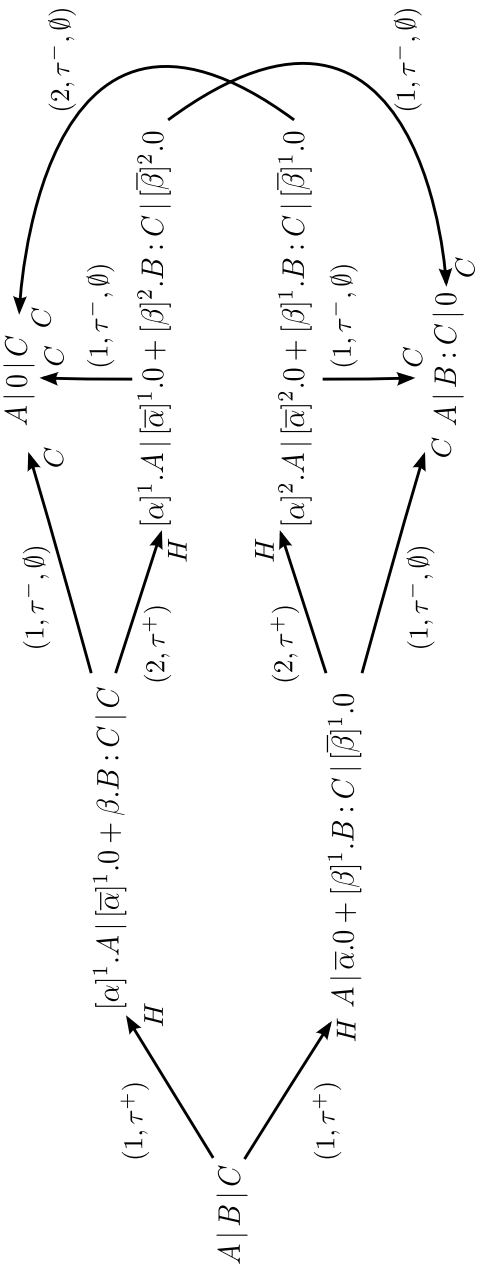

FIGURE 8.4: The LTS for the CCSP 2-reactions model.

Chapter 9

Bio-PEPA models with delay

In CCS it is adopted a modeling approach sometimes referred to as *processes-as-molecules*. A consequence of that is that we require configurations in which unique identifiers are assigned to an instance of synchronization. There are some algebras such as Bio-PEPA (Ciocchetta & Hillston, 2008; Ciocchetta & Hillston, 2009) in which the modeling approach is referred to as *processes-as-species*. In the context of systems biology, such an approach is more comfortable since this may result in more compact models. Moreover, stochastic semantics can be defined more easily.

In this chapter we extend the well-known stochastic process algebra Bio-PEPA to actions with delay. We start recalling the syntax and the semantics of Bio-PEPA, and we discuss the modeling of non-delayed biological systems in such a language. Moreover, we discuss analysis techniques for Bio-PEPA models.

Then we extend the definition of Bio-PEPA to support actions with delays following the delay-as-duration approach. This yields to the definition of a new non-Markovian stochastic process algebras. For such a new algebra we present similar analysis techniques as the one discussed for Bio-PEPA. We also investigate its relationship with Bio-PEPA, both at the level of the semantics and the probabilities involved.

We end the chapter with discussion on the encoding of the models discussed in Chapter 3 in Bio-PEPA with delays.

9.1 Bio-PEPA

In the following sections we formally introduce Bio-PEPA (Ciocchetta & Hillston, 2008; Ciocchetta & Hillston, 2009). We start with some intuitions about the language and then we formally define the syntax of processes and systems. We present a semantics of the language and we discuss analysis techniques for Bio-PEPA models.

Introduction

Bio-PEPA (Ciocchetta & Hillston, 2009) is a *stochastic process algebra*, based on the *Performance Evaluation Process Algebra* (PEPA) (Hillston, 1996), for the modeling and the analysis of biochemical networks. Bio-PEPA models can be considered as intermediate, formal, compositional representation of biological systems, and can be analyzed by means of different analysis techniques. So for instance they can be translated either in deterministic or stochastic models, in the former case they are translated in ODEs, in the latter in either CTMCs, input for the SSA or input for the PRISM model checker (Kwiatkowska *et al.*, 2007; Heath *et al.*, 2008).

In Bio-PEPA special purpose biologically-inspired operators are defined to make modeling of biological systems easier than it is by using different purpose languages. So for instance in Bio-PEPA it is quite straightforward to describe models with reactions having arbitrary stoichiometry

values and following general general kinetic laws. Bio-PEPA supports systems with *static compartment structure* since this permits to have a syntax of the language easy but expressive enough to describe most of the features of the biochemical networks. Moreover, this assumption is also supported by the fact that information present in the literature and in specialized databases (Le Novére *et al.*, 2006) about compartments is often lacking.

A peculiar characteristic of Bio-PEPA is the introduction of an abstraction based on the notion of *discrete concentration level* within a species. In this sense, each species turns out to be parametric with respect to some concentration level. The reason for this feature, which can not be found in any other process algebra for the modeling of biological systems is tackling the problem of incomplete information in the exact number of elements. A direct consequence of this feature is a *reduction of the state space* which helps in facing the computational hardness of model analysis.

We discuss a bit more in details the level feature in Bio-PEPA. To each level is associated a concentration interval and to the whole system a *step size* $H \in \mathbb{N}$, representing its granularity is assigned. In this sense, Bio-PEPA models are quantitatively deÞned in terms of concentrations, but can be expressed terms of levels where these are a discretization of the concentration. The use of the step size H is such that changing the level concentration of a species by one, implies a change in H units of concentration of that species. In this sense, this value can be thought as the level of detail of the considered model, where $H = 1$ is the most detailed model. Since all the species are associated to the same step size H, then in accordance with the law of conservation of mass the concentrations between consumed reactants and the created products is balanced. As we said, to each species a maximum finite concentration level is assigned so that finiteness of the state space strengthens feasibility of model analysis. If we denote with M_i the maximum concentration level for species i, and with x_i its current concentration, then the discrete concentration level for the species is a value $\lceil x_i/H \rceil$ such that

$$0 \leq \left\lceil \frac{x_i}{H} \right\rceil \leq \left\lceil \frac{M_i}{H} \right\rceil .$$

This simple relation implies that to each species a set of $\lceil M_i/H \rceil + 1$ distinct concentration levels can be associated, moreover it implies that in Bio-PEPA negative populations are not allowed. Notice that when $H = 1$ the discrete concentration level reduces to $0 \leq x_i \leq M_i$ and sometimes we term this scenario as a Bio-PEPA model with the *explicit number of molecules*.

9.1.1 Processes and systems

In this section we present the syntax of Bio-PEPA as originally defined in (Ciocchetta & Hillston, 2009). A model is described by *sequential components* representing species, and by some *model components* representing their possible interactions.

We assume an infinite set of action types $\mathcal{A} = \{\alpha, \beta, \gamma, \ldots\}$ and we start by recalling the syntax of the processes.

Definition (Bio-PEPA Processes) Bio-PEPA *processes* are defined by the following grammar:

$$\begin{aligned} S &::= (\alpha, \kappa) op\, S \quad | \quad S + S \quad | \quad C \\ P &::= P \underset{\mathcal{L}}{\bowtie} P \quad | \quad S(l) \end{aligned}$$

where $\alpha \in \mathcal{A}$, $op \in \{\downarrow, \uparrow, \odot, \oplus, \ominus\}$, $\mathcal{L} \in \wp(\mathcal{A})$ is a set of actions and $l, \kappa \in \mathbb{N}$. We denote with $\mathcal{S}$ the set of all possible species specifications, and we denote with $\mathcal{P}$ the set of all possible well-formed Bio-PEPA processes.

The components S and P represent species and their possible interactions, respectively. The element C is used to define constant processes with the usual notation $C \stackrel{def}{=} P$. Elements from S are named sequential components, elements from P are named model components. A notion of well-formed processes is necessary to ensure that a species consists of a choice between reactions, and no reaction name α is repeated within a species. At the model level, there can only be one

species component for each species, as intuitive. As in (Ciocchetta & Hillston, 2009) well-formed processes are those satisfying the following constraints.

Definition (Well-Formedness) A sequential component C is *well-formed* if it has the form

$$C \stackrel{def}{=} (\alpha_1, \kappa_1)op_1 C + \ldots + (\alpha_n, \kappa_n)op_n C$$

where $i \neq j \implies \alpha_i \neq \alpha_j$. A model component P is *well-formed* if it has the form

$$P \stackrel{def}{=} C_1(l_1) \underset{\mathcal{L}_1}{\bowtie} \ldots \underset{\mathcal{L}_n}{\bowtie} C_n(l_n)$$

and each C_i is well-formed, the elements of $\mathcal{L}_i$ appear in P and $i \neq j \implies C_i \neq C_j$.

Bio-PEPA actions are used to model the events (i.e. the reactions) happening in the biological systems we model. The prefix terms in this algebra contain information about the role of the species in the actions. In particular, for $(\alpha, \kappa)op\ S$ we have that (α, κ) is the prefix, where $\alpha \in \mathcal{A}$ is the action type and κ is the stoichiometry coefficient of the species in the reaction. The prefix combinator *"op"* represents the role of the species in the reaction. In particular, $\downarrow$ indicates a reactant, $\uparrow$ a product, $\oplus$ an activator, $\ominus$ an inhibitor and $\odot$ a generic modifier. The actions can appear in a summation term $S_1 + S_2$, whose meaning is the classical "choice" of process algebras.

As we said, in Bio-PEPA a discrete concentration level l is associated with each species, and this is denoted by $S(l)$. We recall that l ranges over $\{0, \ldots, M_S\}$ where M_S is the maximum level of concentration for S to bound the population size, as we discussed in the previous section.

Bio-PEPA supports multiway synchronization which makes easy to model n-ary reactions, whose modeling in dyadic process algebras is not trivial. The term $P_1 \underset{\mathcal{L}}{\bowtie} P_2$ denotes cooperation between P_1 and P_2 over the *cooperation set* $\mathcal{L}$, which determines those activities on which the cooperands are forced to synchronize. For action types not in $\mathcal{L}$, the components proceed independently and concurrently with their enabled activities.

Notice that in Bio-PEPA processes no kinetic information are represented. In fact, processes only model interactions with stoichiometry as quantitative information, but without any information on the rate of the interactions. To this extent, the notion of process is to be extended; a Bio-PEPA model specification is given in terms of *systems* defined as follows.

Definition (Bio-PEPA Systems) A Bio-PEPA *system* is a 6-tuple $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P \rangle$ where:

- $\mathcal{V}$ is the set of *compartments*;
- $\mathcal{N}$ is the set of *quantities* describing each species;
- $\mathcal{K}$ is the set of *parameter* definitions;
- $\mathcal{F}$ is the set of *functional rate* definitions;
- $Comp$ is the set of *sequential component* definitions;
- P is the *initial process* definition.

We denote the set of all Bio-PEPA systems as $\mathcal{R}$.

In Bio-PEPA the kinetic characteristics of the actions are not specified in the syntax of processes as in other calculi but, instead, they are separately represented in the notation of system. This makes in general easy to change the kinetic information of a system, without replacing the process modeling the reactions which can happen in the system.

We briefly discuss the definition of system. If the model assumes more than one compartment, then all the compartment are listed in $\mathcal{V}$, together with information on the compartment volume. In this thesis, we only consider models with a single compartment, and hence we never discuss about $\mathcal{V}$. The maximum concentration levels for each species, as well as the fixed step size H are parameters stored in $\mathcal{N}$. In the definition of system the information about rates is given in $\mathcal{F}$ and

that about kinetic constants is given in $\mathcal{K}$. In any case, all the sequential components are declared in the set *Comp*, and possibly used inside P, the initial process definition.

We spend a few words on the definition of the functional rates, namely the element of $\mathcal{F}$. In Bio-PEPA arbitrary rate functions can be defined, and in (Ciocchetta & Hillston, 2009) the syntax of the rate expression to define rate functions is given. For the sake of helping modelers, some rate functions are already defined in Bio-PEPA. In particular, the predefined kinetic laws are the ones used more frequently: mass action, denoted as f_{MA}, Michaelis-Menten, denoted as f_{MM} and Hill kinetics, denoted as f_H. All these functions depend on some parameters as the components or species involved, which are derived from the context. Moreover, in the functional rates some parameter constants can be used, if this is the case they must be defined in the set $\mathcal{K}$.

A notion of well-formed Bio-PEPA system is required. Intuitively, a system is well-formed if all the quantities involved are precisely declared in the correct sets, and if P is well-formed. For the further definitions and explanations of the components of a Bio-PEPA system, as well as for the definition of well formed system, we again refer to (Ciocchetta & Hillston, 2009).

9.1.2 A Structural Operational Semantics for Bio-PEPA

To Bio-PEPA is given by a Structural Operational Semantics (SOS) (Plotkin, 1981) based on a *capability relation* which supports the derivation of quantitative information and which is auxiliary to a *stochastic relation*. The stochastic relation associates the rates with the actions performed. The use of two relations allows for the association of the rate with the last step of the derivation representing a given reaction, which makes it easier to derive the rate in the appropriate way, especially in the case of general kinetic laws different from mass-action. We now present and comment the SOS of Bio-PEPA.

The capability relation. The capability relation for Bio-PEPA is the one which makes a process move by performing some of its enabled actions. In this sense, such a relation defines all the possible interactions and provides information to compute stochastic information about the performed interactions. The capability relation of Bio-PEPA is $\rightarrow_c \subseteq \mathcal{P} \times \Theta \times \mathcal{P}$ where

$$\Theta = \{(\alpha, w) \mid \alpha \in \mathcal{A}, w \in W\}$$

and W is the set of lists defined by the following grammar

$$w \ ::= \ [S : op(l, \kappa)] \quad | \quad w@w$$

where $S \in \mathcal{S}$, $op \in \{\downarrow, \uparrow, \odot, \oplus, \ominus\}$, $l, \kappa \in \mathbb{N}$, and @ the classical concatenation operator on lists (Milner *et al.*, 1990). The meaning of the parameters in these lists are the ones expected: κ is a stoichiometry value, l is a level, α is an action and S is species.

The rules defining the capability relation are given in Table 9.1. Formally, in rule `(PrefixReac)` a species $((\alpha, \kappa)\downarrow S)(l)$ is involved as reactant in an action, and its concentration level is decreased by κ. Differently, in the case of a species involved as a product, as in rule `(PrefixProd)`, its concentration level is increased by κ. In all the other cases, as in rules `(PrefixAct)` and `(PrefixOp)`, the concentration level is left unchanged. These rules have some contraints to be applied which are induced by the definition of levels. For actions in which it is assumed to have at least κ reactants, namely when a species is involved as a reactant or an activator, the constraints requires $\kappa \leq l \leq N$ if N is the maximum level for the species. Similar constraints are defined for the other cases; notice for instance that for products we require to have space left for at least κ molecules in the concentration level. Notice also that in the case of a reactant the constraint corresponds to the applicability condition of the reaction.

Whenever a transition is derived from any of these processes, the labels exhibited are the action performed, α, and a list containing a single element $S : op(l, \kappa)$. The list contains information about the species which performed the action, S, the role of the species in the action op, the current level of concentration l and the stoichiometry of the species involved in α, κ. It is fairly easy to notice

$$\texttt{(PrefixReac)} \quad \frac{\kappa \leq l \leq N}{((\alpha,\kappa)\downarrow S)(l) \xrightarrow{(\alpha,[S:\downarrow(l,\kappa)])}_c ((\alpha,\kappa)\downarrow S)(l-\kappa)}$$

$$\texttt{(PrefixProd)} \quad \frac{0 \leq l \leq N-\kappa}{((\alpha,\kappa)\uparrow S)(l) \xrightarrow{(\alpha,[S:\uparrow(l,\kappa)])}_c ((\alpha,\kappa)\uparrow S)(l+\kappa)}$$

$$\texttt{(PrefixAct)} \quad \frac{\kappa \leq l \leq N}{((\alpha,\kappa) \oplus S)(l) \xrightarrow{(\alpha,[S:\oplus(l,\kappa)])}_c ((\alpha,\kappa) \oplus S)(l)}$$

$$\texttt{(PrefixOp)} \quad \frac{1 \leq l \leq N}{((\alpha,\kappa)opS)(l) \xrightarrow{(\alpha,[S:op(l,\kappa)])}_c ((\alpha,\kappa)opS)(l)} \text{ if } op = \odot, \ominus$$

$$\texttt{(Choice}_1\texttt{)} \quad \frac{S_1(l) \xrightarrow{(\alpha,w)}_c S_1'(l')}{(S_1+S_2)(l) \xrightarrow{(\alpha,w)}_c S_1'(l')}$$

$$\texttt{(Choice}_2\texttt{)} \quad \frac{S_2(l) \xrightarrow{(\alpha,w)}_c S_2'(l')}{(S_1+S_2)(l) \xrightarrow{(\alpha,w)}_c S_2'(l')}$$

$$\texttt{(Constant)} \quad \frac{S(l) \xrightarrow{(\alpha,w)}_c S'(l') \quad C \stackrel{def}{=} S}{C(l) \xrightarrow{(\alpha,w)}_c S'(l')}$$

$$\texttt{(Coop}_1\texttt{)} \quad \frac{P_1 \xrightarrow{(\alpha,w)}_c P_1' \quad \alpha \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha,w)}_c P_1' \underset{\mathcal{L}}{\bowtie} P_2}$$

$$\texttt{(Coop}_2\texttt{)} \quad \frac{P_2 \xrightarrow{(\alpha,w)}_c P_2' \quad \alpha \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha,w)}_c P_1 \underset{\mathcal{L}}{\bowtie} P_2'}$$

$$\texttt{(Coop}_3\texttt{)} \quad \frac{P_1 \xrightarrow{(\alpha,w_1)}_c P_1' \qquad P_2 \xrightarrow{(\alpha,w_2)}_c P_2' \quad \alpha \in \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha,w_1 \circledcirc w_2)}_c P_1' \underset{\mathcal{L}}{\bowtie} P_2'}$$

TABLE 9.1: The BIO-PEPA capability relation $\rightarrow_c \subseteq \mathcal{P} \times \Theta \times \mathcal{P}$.

that this information are all needed to evaluate information on the kinetics at which this action has been performed.

Rule (Constant) models the move of a constant process, if its associated process is able to move. Rule (Choice$_1$), the symmetric of (Choice$_2$), performs a choice among two sequential components, as in classical process algebras (e.g. see the CCS choice operator we discussed in SECTION 8.1).

Finally, rules (Coop$_1$), (Coop$_2$) and (Coop$_3$) combine an action with the cooperation operator. If two processes are able to cooperate, namely share an action in their cooperation set, then they perform the same action α, as in rule (Coop$_3$), and then the lists they exhibit are concatenated. Notice that the ordering of concatenation is not imposed by any constraint, and in the resulting list all the information about the species in P_1 and P_2 which performed α is contained. Similarly the others rule model the independent move of one of the two processes, once they do not share the action in the cooperation set.

The stochastic relation. As we said, the capability relation supports the derivation of quantitative information and is auxiliary to a stochastic relation able to reflect the action at the level of systems, and to enrich the exhibited labels with kinetic information on the performed action. The

$$\texttt{(Stoch)} \quad \frac{P \xrightarrow{(\alpha,w_1)}_c P' \qquad r_\alpha = f_\alpha[w,\mathcal{N},\mathcal{K}]H^{-1}}{\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,P\rangle \xrightarrow{(\alpha,r_\alpha)}_s \langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,P'\rangle}$$

TABLE 9.2: The BIO-PEPA stochastic relation $\rightarrow_s \subseteq \mathcal{R} \times \Gamma \times \mathcal{R}$.

stochastic relation is $\rightarrow_s \subseteq \mathcal{R} \times \Gamma \times \mathcal{R}$ where

$$\Gamma = \{(\alpha, r_\alpha) \mid \alpha \in \mathcal{A}, r_\alpha \in \mathbb{R}^+\}\,.$$

The rule defining such a relation is given in TABLE 9.2. The stochastic relation associates the rates with the actions performed, in fact r_α represents the parameter of an exponential distribution and, as expected, all activities enabled attempt to proceed but only the fastest succeeds. The rate of any action is computed as

$$r_\alpha[w,\mathcal{N},\mathcal{K}] = f_\alpha[w,\mathcal{N},\mathcal{K}]H^{-1}\,.$$

For the formal explanation of how the rates are computed because of the levels we refer to (Ciocchetta & Hillston, 2009). Intuitively, the notation $f_\alpha[w,\mathcal{N},\mathcal{K}]$ means that the function f_α is evaluated over w, $\mathcal{N}$ and $\mathcal{K}$ so that for each component C_i its concentration l_i is derived and properly combined by means of its correct rate expression with the step size H and the stoichiometry κ.

Finally, we can define BIO-PEPA *semantics* by means of a LTS which, in (Ciocchetta & Hillston, 2009) is termed *stochastic* LTS since its labels contain stochastic information about the transitions.

Definition (BIO-PEPA Semantics) The *semantics* of BIO-PEPA is given by the LTS $(\mathcal{R}, \Gamma, \rightarrow_s)$ where $\rightarrow_s$ is the minimal relation satisfying the rule given in TABLE 9.2.

In (Ciocchetta & Hillston, 2009) some equivalences relations for BIO-PEPA processes and systems are defined. In particular, equivalence relation based on *isomorphisms* of LTSs and bisimulations are investigated. The former is a very strict equivalence notion which relates processes and systems with the same underlying LTS. The latter is the bisimulation we discussed in the previous chapters, enriched considering stochastic feature represented in the LTS.

However, as we already said when introducing general bisimulation, we require more complex equivalence notions inspired by context of application of biological systems. So for instance in (Galpin & Hillston, 2011) a notion of *compression bisimulation* is introduced. Such an equivalence is directly related to the notion of levels, the step size and the maximum number of levels and is able to relate two systems differing only in the step size. Moreover, such an equivalence is a congruence with respect to the synchronization operator.

A BIO-PEPA toy example

In order to clarify the modeling with BIO-PEPA we present a toy example of a model. We assume to model a transformation event from one element of species A to one element of species B. Transformation happens at a rate k and obeys a mass-action kinetic law. Such a model is constituted by a single reaction of the form $A \overset{k}{\mapsto} B$. The initial state contains three elements of species A and no elements of species B; algebraically it is described by the 2-dimensional vector $\mathbf{x}_0 = (3,0)^T$.

The BIO-PEPA processes modeling the species can be easily defined as follows:

$$A \overset{def}{=} (\alpha,1)\downarrow A \qquad B \overset{def}{=} (\alpha,1)\uparrow B$$

where α is the action which models the reaction. Species A is involved in action α as a reactant with stoichiometry 1. Differently, species B is involved as a product with the same stoichiometry

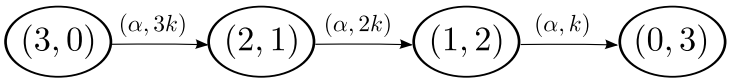

FIGURE 9.1: The LTS for the BIO-PEPA toy example.

of A. The functional rates are defined according to the mass action kinetics, namely by defining $f_\alpha = f_{MA}(k)$. The BIO-PEPA process describing the interacting components is $A \underset{\{\alpha\}}{\bowtie} B$ which represents the fact that the two processes representing the species synchronize to perform action α, the transformation.

By considering levels we assume the species to have some maximum levels N_A and N_B where $N_A > 3$ and $N_B > 3$. The initial levels of concentrations are described by the vector $\mathbf{x}_0$, and the initial BIO-PEPA process is the following

$$A(3) \underset{\{\alpha\}}{\bowtie} B(0)\,.$$

We discuss now on the possible transitions which can be derived by such a process by means of the capability and the stochastic relations. Initially, we have the capability derivations

$$A(3) \xrightarrow{(\alpha,[A:\downarrow(3,1)])}_c ((\alpha,1)\downarrow A)(2) \qquad\qquad B(0) \xrightarrow{(\alpha,[B:\uparrow(0,1)])}_c ((\alpha,1)\uparrow B)(1)$$

which compose as

$$A(3) \underset{\{\alpha\}}{\bowtie} B(0) \xrightarrow{(\alpha,[A:\downarrow(3,1)]@[B:\uparrow(0,1)])}_c A(2) \underset{\{\alpha\}}{\bowtie} B(1)\,.$$

From this derivation, for a generic system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(3) \underset{\{\alpha\}}{\bowtie} B(0)\rangle$ where the kinetic information for this model are considered, we have the stochastic derivation

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(3) \underset{\{\alpha\}}{\bowtie} B(0)\rangle \xrightarrow{(\alpha,3k)}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(2) \underset{\{\alpha\}}{\bowtie} B(1)\rangle\,.$$

Similarly, we can observe capability derivations from which stochastic derivations can be built as follows

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(2) \underset{\{\alpha\}}{\bowtie} B(1)\rangle \xrightarrow{(\alpha,2k)}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(1) \underset{\{\alpha\}}{\bowtie} B(2)\rangle$$
$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(1) \underset{\{\alpha\}}{\bowtie} B(2)\rangle \xrightarrow{(\alpha,k)}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, A(0) \underset{\{\alpha\}}{\bowtie} B(3)\rangle\,.$$

For this system no other stochastic derivations are possible. In fact, we discussed the unique possible evolution of the system and the obtained LTS, as expected, is finite. A graphical representation of the state-transitions for the process is given in Figure 9.1. In that figure, all the states are represented as circles where the notation (n_1, n_2) represents the discrete levels of concentration of the species A, n_1, and B, n_2. All the arrows represent stochastic derivations of the whole system, where the labels are exactly those computed by that relation.

As expected, this system, starting from the initial configuration $\mathbf{x}_0$, namely state $(3, 0)$, eventually reaches the final state $(0, 3)$, which corresponds to the final configuration $A(0) \underset{\{\alpha\}}{\bowtie} B(3)$ and to the vector $(0, 3)^T$.

9.1.3 Analysis techniques in BIO-PEPA

An important reason for using BIO-PEPA is that multiple analysis techniques are possible for the systems we defined. In fact, as we said BIO-PEPA systems are considered as intermediate, formal, compositional representation of the biological model. In this sense, BIO-PEPA systems can be analyzed as CTMCs with levels, can be translated in sets of ODEs, can be translated in an input of the SSA and, finally, can be model checked by means of PRISM (Kwiatkowska *et al.*, 2007). In the following, we briefly discuss the intuitions on all of the above techniques; for a detailed exposition we refer to (Ciocchetta & Hillston, 2009).

From Bio-PEPA systems to ODEs. When defining a deterministic ODE model of a system we need to identify the involved components, and from the event involving the components we then define the terms in the equations. Similarly, a Bio-PEPA system is defined by means of its components, and from the events involving such components the interacting processes are defined. In this sense, it is fairly intuitive the relation between Bio-PEPA systems and ODEs.

In fact, in (Ciocchetta & Hillston, 2009) an algorithm for encoding a generic system in a set of ODEs is defined. The algorithm derives a stoichiometry matrix for the input system starting from the syntactic definition of the initial process. Intuitively, the entries in such a matrix are obtained by analyzing all the system components, all the actions which can be performed and the role of the components in the action. This information is then augmented by considering kinetic properties of the input system, leading to the definition of a kinetic law vector. Such a vector maps the functional rates of the actions to corresponding algebraic terms for the ODEs. Finally, the deterministic variables are associated with the components and an ODEs system is generated. Once that from the initial process are obtained the initial concentrations by means of the input initial levels the system can be numerically analyzed.

Bio-PEPA systems and CTMCs. To each Bio-PEPA system a CTMC with levels can be associated. To have this intuition is enough to consider the LTS of the Bio-PEPA toy example discussed in the previous section: the graphical representation of the LTS precisely reminds of a CTMC. In fact, in (Ciocchetta & Hillston, 2009) a precise characterization of such correspondence is given. The terminology “with levels" means that the values in the states of the chain represent the levels of the processes involved and the transitions from one state to another describe some variations in these levels. This give intuition to the fact that the stochastic derivations must be somehow related to such transitions. Results obtained permit to consider the LTS of a Bio-PEPA system as a CTMC. Moreover, the use of bounds on the maximal concentration levels makes the LTS finite, and hence the CTMC finite as well. In fact, to Bio-PEPA models can be applied the techniques deployed for the analysis of finite CTMCs, for instance steady state distribution can be computed.

Bio-PEPA systems and the SSA. Bio-PEPA actions are instantaneous and a system can be described as a vector in a proper vector space, as in the encoding to CTMCs, consequently Bio-PEPA systems are candidate to be simulated by the SSA. In fact, in (Ciocchetta & Hillston, 2009) the translation of a Bio-PEPA system to SSA input is given similarly to the encoding into ODEs. In such an encoding, the initial number of molecules for a species is calculated as a function of the concentration, some kinetic parameters for the system and the Avogadro number, namely the number of molecules in a mole of a substance. It is important to recall that the SSA assumes propensity functions defined in a specific analytical form, as described in SECTION 2.4.1. As a consequence, since Bio-PEPA supports arbitrary rate functions and hence it describes a wider class of models than the one which can be simulated by the SSA. In this sense, the translation as input of the SSA of systems using only mass action kinetics is guaranteed to be correct. Differently, in more complex cases it is up to modeler verify the applicability of the analysis by means of SSA simulations.

Bio-PEPA systems and PRISM. Probabilistic model checking of Bio-PEPA systems is possible by using PRISM (Kwiatkowska *et al.*, 2007), a tool for formal modeling and analysis of probabilistic systems. On PRISM systems it is possible to specify quantitative properties using the *temporal logic* named *Continuous Stochastic Logic* (Aziz *et al.*, 1996; Baier *et al.*, 1999). The underlying mathematical model of systems specified in PRISM are CTMCs, and in (Ciocchetta & Hillston, 2009) Bio-PEPA systems are mapped to PRISM models where the variables of PRIMS express levels of concentration. Since model checking is beyond the scope of this thesis, we do not further discuss model checking of Bio-PEPA systems. For a detailed discussion the reader is referred to (Ciocchetta & Hillston, 2009).

9.2 Bio-PEPAd: Bio-PEPA with Delays

In the previous sections we recalled the definition of Bio-PEPA , in this section we define an extension of Bio-PEPA where the actions can have a delay. The possibility of assigning delays to actions yields to the definition of a new non-Markovian process algebra: Bio-PEPA *with delays* (Bio-PEPAd). For the sake of simplifying the exposition we consider only delays following the delay-as-duration approach.

Bio-PEPAd is based on the same syntax as Bio-PEPA, hence the definition of Bio-PEPAd systems with delays can be easily obtained by adding, to a Bio-PEPA system of the target model, the delay specifications. A consequence of being based on Bio-PEPA is that Bio-PEPAd contains two aspects to tackle model reduction: the use of the level of concentrations for the species, as in Bio-PEPA, and the delays, as a new feature.

As in CCSd and CCSp the semantics of the Bio-PEPAd is given in the style of the ST-semantics (Bravetti *et al.*, 1998), and the clear logical relationship between Bio-PEPA and Bio-PEPAd gives us the opportunity to prove theorems on the correspondence between the semantics of the two languages and the relation between the mathematical structures underlying the models.

Following the results on Bio-PEPA analysis, we outline how to encode Bio-PEPAd systems in Gsmps, how to perform stochastic simulation of Bio-PEPAd systems using the DDa and how to automatically translate a system in a set of DDes.

As far as applications of Bio-PEPAd are concerned, we encode into Bio-PEPAd the model of the cell-cycle with delays we studied in the previous chapters. Moreover, we encode into Bio-PEPAd the target models presented in Section 3.1.1-3.1.3.

9.2.1 Process configurations and systems

The fact that in Bio-PEPA the kinetic information of a model is separately represented from the Bio-PEPA process gives us the opportunity to use the syntax of Bio-PEPA processes in the context of Bio-PEPAd.

Indeed, processes of Bio-PEPAd are defined by the same syntax as Bio-PEPA processes, so that it is possible to easily extend a Bio-PEPA system into one with delays. Consequently, the delays, which are also properties of the actions which can be performed, are similarly represented separately from processes. Indeed, they are defined by functions belonging to the family

$$\Big\{\sigma : \mathcal{A} \mapsto \{r \in \mathbb{R} \mid r > 0\}\Big\} \in \Delta$$

such that $\sigma(\alpha)$ denotes the delay of action $\alpha \in \mathcal{A}$. From the biological perspective, the choice of using a function σ to specify the delays implies that, for every participant in an action α, a unique delay $\sigma(\alpha)$ corresponds, which is sound since for each species involved in the reaction modeled by α the delay is unique, as in all the DSSAs we presented.

For the sake of simplicity we assume all the actions to have a non-zero delay, the combination of delayed and non-delayed actions can be defined in a natural way by merging the results we present together with those presented in (Ciocchetta & Hillston, 2009) and recalled in the previous sections.

A Bio-PEPAd system is defined as an extension of a Bio-PEPA one as follows.

Definition (Bio-PEPAd Systems) A Bio-PEPAd system is a 7-uple $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P\rangle$ where:

- $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle \in \mathcal{R}$ is a Bio-PEPA system;
- $\sigma \in \Delta$ is a function used to specify the delays of the actions.

We denote with $\tilde{\mathcal{P}}$ the set of all possible well-formed Bio-PEPAd systems.

A Bio-PEPAd system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P\rangle$ is well-formed if is embedded Bio-PEPA system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle$ is well-formed. Again, moving from a Bio-PEPA system specification to a

Bio-PEPAd one is straightforward. This permits us, in the future, to reuse the system specifications for Bio-PEPA in the context of Bio-PEPAd.

Notice that, as in CCSd and CCSp a clear distinction between processes in which actions are able to start, processes have some started actions and the combination of both is required. As in CCSd and CCSp we introduces a notion of process configuration, logically connected to the notion of process. Let us denote by $\mathcal{D}$ the domain of all the possible tuples of the form (l, κ, α, op), namely

$$\mathcal{D} = \{(l, \kappa, \alpha, op) \mid l, \kappa \in \mathbb{N}, \alpha \in \mathcal{A}, op \in \{\downarrow, \uparrow, \odot, \oplus, \ominus\}\}$$

and with $\mathcal{L}_{\mathcal{D}}$ all the possible lists built over $\mathcal{D}$. For the sake of simplicity, we do not give a set-definition of $\mathcal{L}_{\mathcal{D}}$. We define a notion of *process configuration* for Bio-PEPAd as follows.

Definition (Bio-PEPAd Process Configurations) *Process configurations* of Bio-PEPAd are defined by the following syntax:

$$\begin{array}{rcl} C_S & ::= & (\alpha, \kappa) op\, C_S \quad \mid \quad C_S + C_S \quad \mid \quad C \\ C_P & ::= & C_P \underset{\mathcal{L}}{\bowtie} C_P \quad \mid \quad C_S(l, L) \end{array}$$

where $L \in \mathcal{L}_{\mathcal{D}}$, $l, \kappa \in \mathbb{N}$, $\alpha \in \mathcal{A}$ and $op \in \{\downarrow, \uparrow, \odot, \oplus, \ominus\}$. We denote with $\mathcal{C}$ the set of all well-formed process configurations.

The notion of well-formed process configuration is straightforward: any process configuration is well-formed if, by removing the list L, its corresponding Bio-PEPA process is well-formed. Notice that, differently from CCSd and CCSp for instance, there is clear syntactic difference between a process and a process configuration. More formally here it does not hold that $\mathcal{P} \subset \mathcal{C}$, however there is a clear logical connection between processes and configurations, as explained now.

For clarity, in the following we denote a generic process configuration as $S(l, L)$. A species $S(l, L)$ is a species with a discrete level of concentration l, like the species $S(l)$ in Bio-PEPA, but which is currently involved in the actions with delay described by the list L. In particular, if the list L contains an entry (l', κ, α, op), this means that there are κ levels of concentration of species S involved in a currently running action α which fired when the discrete level of concentration of species S was l', its role in this instance of the action is described by op. For instance, a species $S(3, [(2, 1, \alpha, \uparrow)])$ is a species with current concentration level 3, involved in a scheduled action α, started when its concentration level was 2, which is going to increase by 1 its concentration level when completed.

Consequently, L is to be considered as a view of the scheduling list used in the DSSAs we previously presented. More precisely, L is a view of only the scheduled events which involve elements of species S.

Another important comparison between Bio-PEPAd process configurations and CCSd process configurations is worth discussing. In these configurations we do not need to assign a unique identifier to an instance of synchronization, as instead it was in CCSd. This is because of the view of processes-as-species of Bio-PEPA despite the processes-as-molecules of CCS. In fact, for any species a unique process maintains locally the information about the instances of the started actions involving such species. In CCS such information is shared among all the possible molecules of the same species, and then CCS requires a mechanism to identify pairs of processes synchronizing.

Another advantage of this view is that it is possible to handle easily this local information of species to have that actions complete in the order in which they start, which was not easy in CCS. Intuitively, the list L by means of its internal ordering of objects can be used as a *First-In First-Out* (FIFO) data structure, as it was for the scheduling lists in the DSSAs. To this extent, it is necessary to define some auxiliary functions for manipulating the scheduling list L in accordance to the considerations we did. We start by defining four functions

$$\begin{array}{ll} \phi : \mathcal{A} \mapsto \mathcal{L}_{\mathcal{D}} \mapsto \mathcal{D} & \qquad \zeta : \mathcal{A} \mapsto \mathcal{L}_{\mathcal{D}} \mapsto \mathcal{L}_{\mathcal{D}} \\ \pi : \mathcal{L}_{\mathcal{D}} \mapsto \mathcal{L}_{\mathcal{D}} & \qquad \rho : \mathcal{L}_{\mathcal{D}} \mapsto \mathbb{N} \end{array}$$

$$\begin{array}{l}\phi\ \alpha\ L\ =\ \underline{match}\ \ L\ \ \underline{with}\\ \qquad |\ [\,]\ \rightarrow \bot;\\ \qquad |\ (l,\kappa,\alpha,op)::xs\ \rightarrow\ (l,\kappa,\alpha,op);\\ \qquad |\ x::xs\ \rightarrow\ \phi\ \alpha\ xs.\end{array}$$

$$\begin{array}{l}\zeta\ \alpha\ L\ =\ \underline{match}\ \ L\ \ \underline{with}\\ \qquad |\ [\,]\ \rightarrow\ [\,];\\ \qquad |\ (l,\kappa,\alpha,op)::xs\ \rightarrow\ xs;\\ \qquad |\ x::xs\ \rightarrow\ x::\zeta\ \alpha\ xs.\end{array}$$

$$\begin{array}{l}\pi\ L\ =\ \underline{match}\ \ L\ \ \underline{with}\\ \quad |\ [\,]\ \rightarrow\ [\,];\\ \quad |\ (l,\kappa,\alpha,\uparrow)::xs\ \rightarrow\ (l,\kappa,\alpha,\uparrow)::\pi\ xs;\\ \quad |\ x::xs\ \rightarrow\ \pi\ xs.\end{array}$$

$$\begin{array}{l}\rho\ L\ =\ \underline{match}\ \ L\ \ \underline{with}\\ \quad |\ [\,]\ \rightarrow\ 0;\\ \quad |\ (l,\kappa,\alpha,op)::xs\ \rightarrow\ \kappa+\rho\ xs.\end{array}$$

TABLE 9.3: Formal functional style definitions of auxiliary functions ϕ, ζ, π and ρ .

whose formal definition is given in TABLE 9.3. In such definitions, we use the classical "cons" operator :: and pattern matching techniques, as described in (Milner *et al.*, 1990).

Function ϕ extracts the first scheduled event with a given action name from the list L; the function value $\phi\,\alpha\,L$ is $\bot$ if no entries of action α exist in L (i.e. no actions α are currently running); otherwise, it is the first entry obtained by a left-to-right recursive scan of L. Notice that we assume the syntactic priority of pattern matching.

Function ζ is used to modify a list such that $\zeta\,\alpha\,L$ is a new list obtained by removing the first, if any, occurrence of an action α obtained by a left-to-right scan of L. As this is an event list, the ordering of insertion of the tuples determines their ordering for extraction. The functions ϕ and ζ, together with the classical concatenation function @, are used to implement a FIFO policy for insertion and extraction of elements in L, as practically required by the DSSAs which schedule reactions. As an example, given a list $T = (2, 1, \alpha, \downarrow) :: l$ the functions are such that

$$\phi\,\alpha\,T = (2, 1, \alpha, \downarrow) \qquad\qquad \zeta\,\alpha\,T = l$$

namely the function ζ removes the entry computed by the function ϕ, when applied with the same parameters.

As in BIO-PEPA we want to keep the state representation of the BIO-PEPAD models finite by using some constraints for the starting of actions. Thus, let us denote the scheduled actions in which the species S is involved as a product by $\pi\,L$. The species $S(l, L)$ is currently involved in the delayed actions as follows: for the scheduled actions in $\pi\,L$ it is involved as a product, and for the other ones it is involved either as a reactant, a modifier, an activator or an inhibitor.

Finally, let us denote by ρ the function computing how many levels of concentration are involved in all the actions described in its input list, regardless of the role of the species in the scheduled event. So for instance, given the list T previously defined, the functions are such that $\pi\,T = \pi\,l$ and $\rho\,T = \rho\,l$ since the first entry of T is discarded.

From these simple considerations general properties relating these functions can be stated and used to help in understanding their use in the definition of the semantics for BIO-PEPAD.

PROPOSITION 9.2.1. *For any $T \in \mathcal{L}_{\mathcal{D}}$ if $T = l'@(l, \kappa, \alpha, op) :: l''$ and the entries in the list l' do not contain α, then the following equations hold*

$$\begin{aligned}\phi\,\alpha\,T &= (l, \kappa, \alpha, op)\\ \zeta\,\alpha\,T &= l'@l''\\ \pi\,T &= (l, \kappa, \alpha, op)@\pi\,l''\\ \rho\,(l, \kappa, \alpha, op) :: l'' &= \rho\,\pi\,T = \kappa + \rho\,l''\,.\end{aligned}$$

By following the DDA in the interpretation of the delays and the stated property this implies that, for species S, there are exactly $\rho\ \pi\ L$ levels of concentration of species S which are currently

waiting for their delay to expire before becoming available in the species S. These two functions can be used to define the constraints to keep the state space finite, as presented in the next sections.

A Bio-PEPAd system specification is typically given in terms of a process $P \in \mathcal{P}$ whose semantics is given in terms of its equivalent process configuration $P_C \in \mathcal{C}$. Intuitively, we want the initial term P to be modified in the corresponding initial configuration P_C where every species declaration $S(l_0, [\,])$ in P_C is such that $S(l_0)$ is in P. The initial process configuration is obtained by adding an empty scheduling list to each species because, in the initial configuration, there are no instances of actions with delay currently running. Formally, we define, by structural recursion on the syntax of processes a function $\mu : \mathcal{P} \mapsto \mathcal{C}$ such that

$$\mu((\alpha,\kappa)op\ S) = (\alpha,\kappa)op\ S \qquad \mu(P_1 \underset{\mathcal{L}}{\bowtie} P_2) = \mu(P_1) \underset{\mathcal{L}}{\bowtie} \mu(P_2)$$
$$\mu(S_1 + S_2) = S_1 + S_2 \qquad \mu(S(l)) = S(l, [\,]).$$

As expected the function is such that the process

$$S(l_1) \underset{\mathcal{L}_1}{\bowtie} S(l_2) \underset{\mathcal{L}_2}{\bowtie} S(l_3)$$

is transformed into the corresponding configuration

$$S(l_1, [\,]) \underset{\mathcal{L}_1}{\bowtie} S(l_2, [\,]) \underset{\mathcal{L}_2}{\bowtie} S(l_3, [\,])$$

where no actions are running.

We augment the definition of Bio-PEPAd systems to 7-tuples of the form

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P_C \rangle$$

where P_C is a process configuration of a process. This is necessary since the semantics of Bio-PEPAd is given in terms of processes which can describe running actions, so which are effectively process configurations, and their generalization into systems.

In the following, we may use the notation P to refer to either a process or a process configuration; it will be clear from the context to which of them we are referring. We denote the extended set of all Bio-PEPAd systems with process configurations as $\overline{\mathcal{P}}$.

9.2.2 A Structural Operational Semantics for Bio-PEPAd

As for CCSd and CCSp in Chapter 8, and similarly to Bio-PEPA where the SOS is defined by means of two relations, in this algebra the SOS, given in a Starting-Terminating (ST) style (Bravetti *et al.*, 1998), is defined by means of three relations. In the following subsections we define a *start relation* on process configurations which, in the same style as the Bio-PEPA capability one, contains the quantitative information needed to evaluate the functional rates and modifies the process configurations to model the start of an action. Also, we define a *completion relation* on process configuration which describes the termination of an action. Finally, along the line of the stochastic relation in Bio-PEPA, we define a *stochastic relation* for Bio-PEPAd systems, based on the start and completion relations, which associates rates with transitions.

As noted earlier, we assume only systems where all the actions are delayed. The SOS we present here can be easily extended to describe a system with both delayed and non-delayed actions.

The start relation

This relation contains the quantitative information to compute rates of starting actions. Also, this relation modifies the process configuration to model the starting of an action, accordingly to the DDA. The start relation is $\rightarrow_{st} \subseteq \mathcal{C} \times \Theta^+ \times \mathcal{C}$ where

$$\Theta^+ = \{(\alpha^+, w) \mid \alpha \in \mathcal{A}, w \in W\}$$

$$\texttt{(StPrefixReac)} \quad \frac{\kappa \leq l \leq N}{((\alpha,\kappa)\downarrow S)(l,L) \xrightarrow{(\alpha^+,[S:\downarrow(l,\kappa)])}_{st} (S)(l-\kappa, L@[(l,\kappa,\alpha,\downarrow)])}$$

$$\texttt{(StPrefixProd)} \quad \frac{0 \leq l+\rho\ \pi\ L \leq N}{((\alpha,\kappa)\uparrow S)(l,L) \xrightarrow{(\alpha^+,[S:\uparrow(l,\kappa)])}_{st} (S)(l, L@[(l,\kappa,\alpha,\uparrow)])}$$

$$\texttt{(StPrefixAct)} \quad \frac{\kappa \leq l \leq N}{((\alpha,\kappa)\oplus S)(l,L) \xrightarrow{(\alpha^+,[S:\oplus(l,\kappa)])}_{st} (S)(l, L@[(l,\kappa,\alpha,\oplus)])}$$

$$\texttt{(StPrefixOp)} \quad \frac{1 \leq l \leq N}{((\alpha,\kappa)op\ S)(l,L) \xrightarrow{(\alpha^+,[S:op(l,\kappa)])}_{st} (S)(l, L@[(l,\kappa,\alpha,op)])} \text{ if } op = \odot, \ominus$$

$$\texttt{(StChoice}_1\texttt{)} \quad \frac{S_1(l,L) \xrightarrow{(\alpha^+,w)}_{st} S_1'(l',L')}{(S_1+S_2)(l,L) \xrightarrow{(\alpha^+,w)}_{st} S_1'(l',L')}$$

$$\texttt{(StChoice}_2\texttt{)} \quad \frac{S_2(l,L) \xrightarrow{(\alpha^+,w)}_{st} S_2'(l',L')}{(S_1+S_2)(l,L) \xrightarrow{(\alpha^+,w)}_{st} S_2'(l',L')}$$

$$\texttt{(StConstant)} \quad \frac{S(l,L) \xrightarrow{(\alpha^+,w)}_{st} S'(l',L') \quad C \stackrel{def}{=} S(l,L)}{C \xrightarrow{(\alpha^+,w)}_{st} S'(l',L')}$$

$$\texttt{(StCoop}_1\texttt{)} \quad \frac{P_1 \xrightarrow{(\alpha^+,w)}_{st} P_1' \quad \alpha \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha^+,w)}_{st} P_1' \underset{\mathcal{L}}{\bowtie} P_2}$$

$$\texttt{(StCoop}_2\texttt{)} \quad \frac{P_2 \xrightarrow{(\alpha^+,w)}_{st} P_2' \quad \alpha \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha^+,w)}_{st} P_1 \underset{\mathcal{L}}{\bowtie} P_2'}$$

$$\texttt{(StCoop}_3\texttt{)} \quad \frac{P_1 \xrightarrow{(\alpha^+,w_1)}_{st} P_1' \quad P_2 \xrightarrow{(\alpha^+,w_2)}_{st} P_2' \quad \alpha \in \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha^+,w_1@w_2)}_{st} P_1' \underset{\mathcal{L}}{\bowtie} P_2'}$$

TABLE 9.4: The BIO-PEPAD start relation $\rightarrow_{st} \subseteq \mathcal{C} \times \Theta^+ \times \mathcal{C}$.

and W is the same set of lists used in BIO-PEPA by the capability relation. The BIO-PEPAD start relation is defined as the minimum relation satisfying the rules presented in TABLE 9.4.

All the rules `(StPrefixReac)`, `(StPrefixProd)`, `(StPrefixAct)` and `(StPrefixOp)` model the start of an action in a species involved accordingly to some role. If a species $((\alpha,\kappa)\downarrow S)(l,L)$ is involved as reactant in an action, then by following the DDA its concentration level is decreased by κ. Differently, in the case of a species involved as a product, its concentration level is not changed because, as previously stated, this relation models the starting and not the completion of an action with delay. In the case of a species taking part in the reaction as a modifier, an inhibitor or an activator, its concentration level is not changed, as expected.

Independently of the role of a species, its scheduling list L is modified to record that some of its levels of concentration are currently performing action α. Notice that, in order to maintain the FIFO property on the scheduling list L, we simply use the append function @. This is possible because of both the multiway synchronization of BIO-PEPAD and the use of fixed deterministic delays, as we previously discussed. Again, two instances of the same action starting in two subsequent instants, are assumed to terminate in two subsequent instants. This is true in a framework where delays are deterministic however, if they were stochastic, the two instances could have multiple orderings for completion. Indeed, because of the multiway synchronization in the scheduling list

L the two instances will appear subsequently and, hence, will complete subsequently. We remark that, in a process calculus with dyadic synchronization, we had to use different timing assumptions to give a reasonably easy semantics.

We use constraints on the levels to have a finite state space as in Bio-PEPA. The constraints for starting the actions are the same as those in Bio-PEPA except the one for the products. In particular, the constraints which must be satisfied by a species $S(l, L)$ to fire an action as a product is, as expected by the previously stated propositions, $0 \leq l + \rho\ \pi\ L \leq N$, if N is its maximum level. Intuitively, this means that the levels of concentration in the state, l, plus those which are already scheduled to be produced, $\rho\ \pi\ L$, must not exceed the capacity threshold N.

The starting of the action α, in the style of the ST semantics, is denoted by the action symbol α^+, exhibited as a label for all the start derivations. The composition of the derivations of this relation is straightforward and almost equivalent to the case of Bio-PEPA, hence we do not comment rules $(\texttt{StChoice}_1)$, $(\texttt{StChoice}_2)$, $(\texttt{StConstant})$, $(\texttt{StCoop}_1)$, $(\texttt{StCoop}_2)$ and $(\texttt{StCoop}_3)$. Notice also that, similarly to CCSd, the choice operator is resolved at the moment of the start of the action, whereas if we had chosen the purely delayed approach, this would have not been possible, as in CCSp.

Some further considerations and comparisons with Bio-PEPA are useful. Firstly, when the actions have no delay as in Bio-PEPA, whenever an action fires, the changes in the process are immediately visible in a one–step derivation, since the Bio-PEPA capability relation modifies the process according to the action. In this algebra, as the instants at which an action starts and terminates are detached, then the start relation modifies the process to represent only the starting of the action. Indeed, another relation, which does not exist in the semantics of Bio-PEPA, models the termination of a currently running action.

Secondly, by comparing the DDa and the definition of this relation, it is clear that the modification of the process to reflect the starting of an action corresponds to scheduling of the reaction in the scheduling list.

The completion relation

This relation is used to model the completion of an action with delay which is currently running. Also, this relation contains quantitative information needed to re-compute the functional rate of the action at the moment in which it started. The completion relation is $\rightarrow_{co} \subseteq \mathcal{C} \times \Theta^- \times \mathcal{C}$ where

$$\Theta^- = \{(\alpha^-, w) \mid \alpha \in \mathcal{A}, w \in W\}$$

and W is the same set of lists used in the start relation. The completion relation is defined as the minimum relation satisfying the rules of Table 9.5.

Rules $(\texttt{CoPrefixProd})$ and $(\texttt{CoPrefixOp})$ model the completion of an action in a species involved accordingly to some role. For any species $S(l, L)$ it is possible to get the instance of a currently running action α, if any, by applying function ϕ. More precisely, this permits us to get, from all the possible instances of actions α, the first which has been scheduled, $\phi\ \alpha\ L$, and, hence, the first completing. If the species is involved as a product, then it is necessary to increase, as defined by the delay-as-duration approach, its concentration level by adding the scheduled products. Otherwise, whatever the role of the species, its concentration level must remain constant, since it has already been updated by the start of the action. Independently of the role of the species in the action, the scheduling list is modified by means of the function ζ, hence a new list $\zeta\ \alpha\ L$ is produced by removing from L the entry which was computed by function ϕ. This last choice is supported by the proposition we previously discussed.

It is unnecessary to state constraints for the completion of a currently running action, as the appropriate ones will have been checked before the action started.

The completion of the action α, in the style of the ST semantics, is denoted by the action symbol α^-, exhibited as a label for all the completion derivations. The other label, namely the list w, is defined like the one exhibited by the start relation. The composition of this relation with the other operators is straightforward and very similar to the composition of the derivations of

$$\texttt{(CoPrefixProd)} \quad \frac{\phi\ \alpha\ L = (l, \kappa, \alpha, \uparrow)}{S(l', L) \xrightarrow{(\alpha^-, [S:\uparrow(l,\kappa)])}_{co} S(l' + k, \zeta\ \alpha\ L)}$$

$$\texttt{(CoPrefixOp)} \quad \frac{\phi\ \alpha\ L = (l, \kappa, \alpha, op)}{S(l', L) \xrightarrow{(\alpha^-, [S:op(l,\kappa)])}_{co} S(l', \zeta\ \alpha\ L)} \text{ if } op = \downarrow, \odot, \oplus, \ominus$$

$$\texttt{(CoChoice}_1\texttt{)} \quad \frac{S_1(l, L) \xrightarrow{(\alpha^-, w)}_{co} S_1'(l', L')}{(S_1 + S_2)(l, L) \xrightarrow{(\alpha^-, w)}_{co} S_1'(l', L')}$$

$$\texttt{(CoChoice}_2\texttt{)} \quad \frac{S_2(l, L) \xrightarrow{(\alpha^-, w)}_{co} S_2'(l', L')}{(S_1 + S_2)(l, L) \xrightarrow{(\alpha^-, w)}_{co} S_2'(l', L')}$$

$$\texttt{(CoConstant)} \quad \frac{S(l, L) \xrightarrow{(\alpha^-, w)}_{co} S'(l', L') \quad C \stackrel{def}{=} S(l, L)}{C \xrightarrow{(\alpha^-, w)}_{co} S'(l', L')}$$

$$\texttt{(CoCoop}_1\texttt{)} \quad \frac{P_1 \xrightarrow{(\alpha^-, w)}_{co} P_1' \quad \alpha \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha^-, w)}_{co} P_1' \underset{\mathcal{L}}{\bowtie} P_2}$$

$$\texttt{(CoCoop}_2\texttt{)} \quad \frac{P_2 \xrightarrow{(\alpha^-, w)}_{co} P_2' \quad \alpha \notin \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha^-, w)}_{co} P_1' \underset{\mathcal{L}}{\bowtie} P_2'}$$

$$\texttt{(CoCoop}_3\texttt{)} \quad \frac{P_1 \xrightarrow{(\alpha^-, w_1)}_{co} P_1' \quad P_2 \xrightarrow{(\alpha^-, w_2)}_{co} P_2' \quad \alpha \in \mathcal{L}}{P_1 \underset{\mathcal{L}}{\bowtie} P_2 \xrightarrow{(\alpha^-, w_1 @ w_2)}_{co} P_1' 1 \underset{\mathcal{L}}{\bowtie} P_2'}$$

TABLE 9.5: The completion relation $\rightarrow_{co} \subseteq \mathcal{C} \times \Theta^- \times \mathcal{C}$.

the start relation. In fact, we do not comment rules $\texttt{(CoChoice}_1\texttt{)}$, $\texttt{(CoChoice}_2\texttt{)}$, $\texttt{(CoConstant)}$, $\texttt{(CoCoop}_1\texttt{)}$, $\texttt{(CoCoop}_2\texttt{)}$ and $\texttt{(CoCoop}_3\texttt{)}$.

Some further considerations are worth noting. Firstly, this relation is a new one with respect to the BIO-PEPA semantics. When actions have no delays this relation would be redundant since all relevant information can be derived from the starting of the action. Probabilistic arguments mean that we can assume that once an exponential action has started the next event will be its completion and there is no need to distinguish the start and completion. When a deterministic delay is associated with an action the role of this relation is to model the completion of an action. To do so it chooses actions to terminate from the list which is associated with the species, namely the list of actions currently running. The start relation, differently, chooses the action to fire from the species definition.

Furthermore, as we want the completion relation to exhibit quantitative information in order to recompute the functional rate of the action at the moment at which it started, then the labels exhibited by this relation are very similar to those exhibited by the start relation, even if they are computed starting from $\phi\ \alpha\ L$. This permits us to have a unique policy for computing the functional rates from the input lists, obtained by derivations of the transitions of these relations.

The stochastic relation

The stochastic relation permits us to associate rates with transitions. Also, this transition permits us to observe changes in a BIO-PEPAD system due to either the starting or the completion of an

$$\texttt{(StochStart)} \quad \frac{P \xrightarrow{(\alpha^+,w)}_{st} P' \qquad r_\alpha = f_\alpha[w,\mathcal{N},\mathcal{K}]H^{-1}}{\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,P\rangle \xrightarrow{(\alpha^+,r_\alpha,\sigma(\alpha))}_{s} \langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,P'\rangle}$$

$$\texttt{(StochCompl)} \quad \frac{P \xrightarrow{(\alpha^-,w)}_{co} P' \qquad r_\alpha = f_\alpha[w,\mathcal{N},\mathcal{K}]H^{-1}}{\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,P\rangle \xrightarrow{(\alpha^-,r_\alpha,\sigma(\alpha))}_{s} \langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,P'\rangle}$$

TABLE 9.6: The stochastic relation $\rightarrow_s \subseteq \overline{\mathcal{P}} \times \Gamma \times \overline{\mathcal{P}}$.

action. The stochastic relation is $\rightarrow_s \subseteq \overline{\mathcal{P}} \times \overline{\Gamma} \times \overline{\mathcal{P}}$ where

$$\overline{\Gamma} = \{(\alpha^*, r_\alpha, \sigma_\alpha) \mid \alpha \in \mathcal{A}, * \in \{+,-\}, r_\alpha, \sigma_\alpha \ \in \mathbb{R}^+\}\,.$$

and is defined as the minimum relation satisfying the rules given in TABLE 9.6.

As in BIO-PEPA, r_α represents the parameter of an exponential distribution and, as expected, all activities enabled attempt to proceed but only the fastest succeeds. Moreover, the third component represent the delay of the action.

As this relation is defined on the set $\overline{\mathcal{P}}$, namely the set of all possible BIO-PEPAD systems with process configurations, whenever we refer to the semantics of a system $\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,P\rangle$, where P is a BIO-PEPA process, we assume we apply the stochastic relation to the system $\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,\mu(P)\rangle$. Again, this is necessary because P is not a process configuration, and we want to build, from P, the corresponding initial configuration $\mu(P)$, and then we want to apply the semantics to the system.

Formally, the starting of an action α, obtained by composition with a derivation of the start relation, is denoted by symbol α^+. The completion of an action is obtained by composition with a derivation of the completion relation, as denoted by symbol α^-. The rate of any action is computed as in BIO-PEPA, namely as $r_\alpha = f_\alpha[w,\mathcal{N},\mathcal{K}]H^{-1}$. For any possible derivation of the stochastic relation, the value $\sigma(\alpha)$ denotes the delay of the action α.

Now we can define BIO-PEPAD semantics as a LTS.

Definition (BIO-PEPAD Semantics) The *semantics* of BIO-PEPAD is given by the LTS $(\overline{\mathcal{P}}, \Gamma, \rightarrow_s)$ where $\rightarrow_s$ is the minimal relation satisfying the rule given in TABLE 9.2.

Even if did not recall any definition of equivalence relations for BIO-PEPA processes, the logical connection between BIO-PEPA and BIO-PEPAD gives intuition on the fact that such equivalences can be easily be defined for BIO-PEPAD.

On timing aspects in BIO-PEPAD. As for CCSD and CCSP, in the LTS we defined there is no explicit notion of time and there are state changes induced by either the start or the completion of an action. In this sense, by the considerations on the scheduling lists, we can consider the duration of an action, namely the time between the start and the completion of an action, as the deterministic delay. As required, two instances of the same action complete in the ordering in which they started.

However, some considerations are worth discussing for both the relations we defined. First of all, the start and the completion relations are by purpose defined over process configurations, not systems. This has the advantage of leaving detached kinetic information about systems from the possible interactions of the processes, as we already discussed. This requires that the start and the completion relations must generate all the possible behaviors for a process configuration and, in fact, this is what effectively is done. However, among all the possible behaviors there could be some which, once the system is really associated to kinetic information, are not physically possible. In this sense, we should filter these derivations by means of the stochastic relation, otherwise the LTS we build is, at a first glance, too rich.

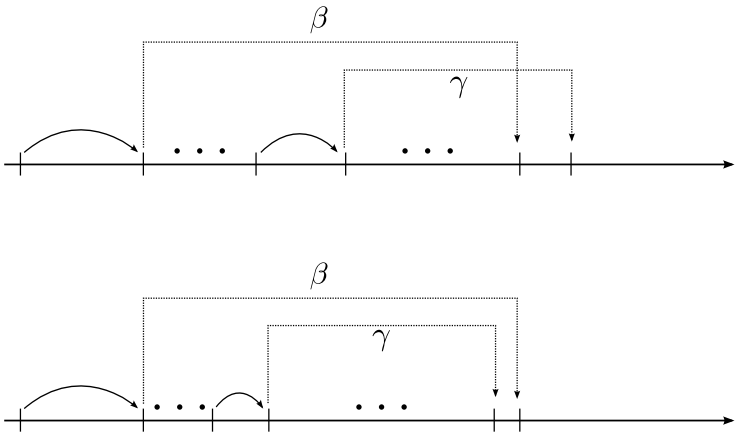

FIGURE 9.2: Timing aspects in BIO-PEPAD.

We clarify this with an informal example. Let us assume that in the process configuration P two actions β and γ are started in such an order. It is easy to notice that the completion relation permits to derive the completion of both the action, $P \xrightarrow{(\beta^-,w_1)}_{co} P'$ and $P \xrightarrow{(\gamma^-,w_2)}_{co} P''$, and by definition the β which completes is the first to complete among all the possibly scheduled β, and the same holds for γ. However, no relation on the order of completion between actions β and γ can be stated. Now, let us consider a possible delays specification such that $\sigma(\beta) > \sigma(\gamma)$. In such a scenario both the derivations are *correct*, in the sense that if we derive P' it means that β completed first, namely γ started and completed across the completion of β, as shown in case (a) of FIGURE 9.2. Differently, in the other case, shown as case (b) of FIGURE 9.2, γ started and completed before the completion of β, and this is possible since $\sigma(\beta) > \sigma(\gamma)$.

Let us consider now the opposite scenario, γ started before β. Among the two derivations we discussed only the one which models the completion of γ is correct. In fact, there is no chance that β completes before γ since $\sigma(\beta) > \sigma(\gamma)$. Hence one of the two derivations should be disregarded.

By generalizing this example, we can notice that in general an action β can complete if all the actions which started earlier than β have delay greater than $\sigma(\beta)$. The stochastic relation is defined over systems, and systems contain information to discover the values of the delays, hence such a relation should not permit to derive transitions which, in the end, never happen. Clearly, the stochastic relation we defined is not precise in this sense, since it does not respect the constraint we informally discussed. However, we have reasons supporting our choice, and we briefly discuss them.

The first is a probabilistic motivation. The aim of BIO-PEPAD systems is to provide specifications of models, whose behavior is precisely described and which can be analyzed by means of some analysis techniques, as it was for BIO-PEPA. The analysis techniques we present in the next sections are correctly supported by the definition of the LTS we gave. In particular, when analyzing the relation between the LTS we defined and a class of non-Markovian stochastic processes it turns out that to the transitions which should filter it is possible to assign probability 0, which practically makes them absent form a probabilistic perspective.

The others motivations are more technical. The second is that the constraint we informally defined should have be defined on the lists of the process, but as in BIO-PEPA it is very convenient to have the scheduling lists to be local for each species. In fact, with respect to the example, it is not possible to establish whether β started before γ, or viceversa, if the two actions do not involve at least one shared species, since in that case the scheduling lists containing the actions are separate and in a timeless semantics the only ordering we have is given by the positions in the scheduling lists.

Finally, if the constraint we require would have been defined on the labels exhibited by the

stochastic derivations of started actions, then the stochastic relation should have been defined on a sequence of steps, which is not what we require since this conflicts with compositional properties of our semantics. Or, differently, we could have stored in the system state the labels exhibited by the start of an action, but this would have been equivalent of having a scheduling list at top-level in the system, which conflicts with the idea of having local scheduling lists.

A Bio-PEPAd toy example

In order to clarify the modeling with Bio-PEPAd we present a simple extension of the Bio-PEPA toy model previously presented. In order to switch to the Bio-PEPAd framework we assume that the reaction $A \overset{k}{\longmapsto} B$, denoting the transformation of an element of species A into an element of species B with a kinetic constant k, is now enriched with a delay σ', giving rise to the definition of the reaction $A \overset{k,\sigma'}{\longmapsto} B$. We assume the initial state described by the vector $\mathbf{x}_0 = (3,0)^T$.

As we defined a conservative extension of Bio-PEPA, we are able to fully reuse the Bio-PEPA specification for this model, namely the process definitions $A \overset{def}{=} (\alpha,1)\downarrow A$, $B \overset{def}{=} (\alpha,1)\uparrow B$, and $A \underset{\{\alpha\}}{\bowtie} B$. Also, the kinetic information about the system is preserved, namely $f_\alpha = f_{MA}(k)$. Conversely, the information about the delay of α, which is not present in Bio-PEPA, is defined according to the function $\sigma(\alpha) = \sigma'$.

By considering the same Bio-PEPA levels, the initial configuration of the process, obtained by applying function μ, is the following

$$A(3,[\,]) \underset{\{\alpha\}}{\bowtie} B(0,[\,]) \equiv S_0\,.$$

We discuss now on the possible transitions which can be derived by such a process by means of the relations we defined. Initially, we have the capability derivations

$$A(3,[\,]) \xrightarrow{((\alpha^+,[A:\downarrow(3,1)]))}_{st} A(2,[(3,1,\alpha,\downarrow)])$$
$$B(0,[\,]) \xrightarrow{((\alpha^+,[B:\uparrow(0,1)]))}_{st} B(0,[(0,1,\alpha,\uparrow)])$$

which compose as

$$A(3,[\,]) \underset{\{\alpha\}}{\bowtie} B(0,[\,]) \xrightarrow{((\alpha^+,[(A:\downarrow(3,1)),(B:\uparrow(0,1))]))}_{st} A(2,[(3,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(0,[(0,1,\alpha,\uparrow)]) \equiv S_1\,.$$

As expected, this configuration contains an instance of α started when the level of A was 3 and the level of B was 0. Such information is also given in the exhibited label. Notice that from the initial process no transition of the completion relation can be derived, as required. From this derivation, for the system $\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,S_0\rangle$ where the kinetic information for this model are considered, we have the stochastic derivation

$$\langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,S_0\rangle \xrightarrow{(\alpha^+,3k,\sigma')}_{s} \langle \mathcal{V},\mathcal{N},\mathcal{K},\mathcal{F},Comp,\sigma,S_1\rangle\,.$$

Now, from S_1, it is possible to observe two different derivations for each species: one corresponding to the completion of the started α, and one corresponding to the start of another instance of α. The former is given by

$$A(2,[(3,1,\alpha,\downarrow)]) \xrightarrow{((\alpha^-,[A:\downarrow(3,1)]))}_{co} A(2,[\,])$$
$$B(0,[(0,1,\alpha,\uparrow)]) \xrightarrow{((\alpha^-,[B:\uparrow(0,1)]))}_{co} B(0,[\,])$$

and as expected the scheduling lists are empty, the latter by

$$A(2,[(3,1,\alpha,\downarrow)]) \xrightarrow{((\alpha^+,[A:\downarrow(2,1)]))}_{st} A(1,[(3,1,\alpha,\downarrow),(2,1,\alpha,\downarrow)])$$
$$B(0,[(0,1,\alpha,\uparrow)]) \xrightarrow{((\alpha^+,[B:\uparrow(0,1)]))}_{st} B(0,[(0,1,\alpha,\uparrow),(0,1,\alpha,\uparrow)])$$

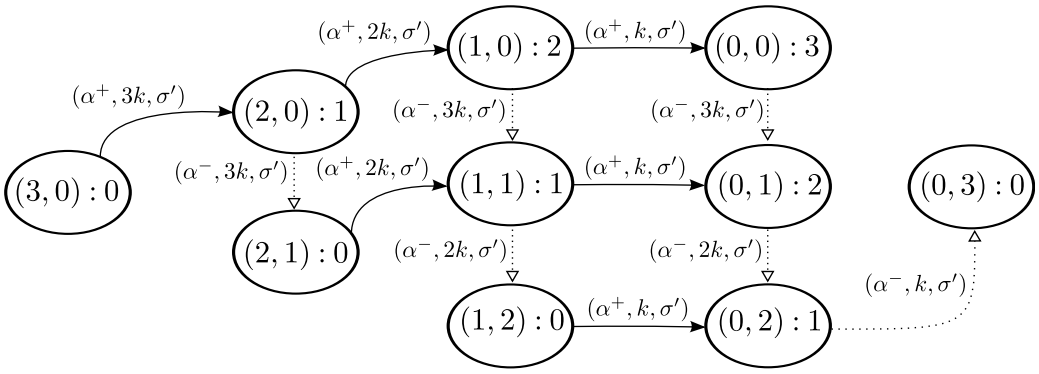

FIGURE 9.3: The LTS for the BIO-PEPAD toy example.

and as expected the latter contains two instances of α not yet completed. Of course, the instances in the list of B are different since they occupy different positions in the list. Once these derivations are composed leading we can derive stochastic transitions as

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, S_1 \rangle \xrightarrow{(\alpha^-, 3k, \sigma')}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, S_2 \rangle$$
$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, S_1 \rangle \xrightarrow{(\alpha^+, 2k, \sigma')}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, S_3 \rangle$$

where

$$S_2 \equiv A(2, [\,]) \underset{\{\alpha\}}{\bowtie} B(1, [\,])$$
$$S_3 \equiv A(1, [(3, 1, \alpha, \downarrow), (2, 1, \alpha, \downarrow)]) \underset{\{\alpha\}}{\bowtie} B(0, [(0, 1, \alpha, \uparrow), (0, 1, \alpha, \uparrow)]) \,.$$

We omit to discuss all the possible derivations of this BIO-PEPAD systems. By applying the stochastic relation to the system with the initial process configuration we obtain all the possible evolutions of the configuration. The obtained LTS, as expected, is finite, and, because of the delays, it corresponds to a non-Markovian stochastic process. Intuitively, there is a one-to-one correspondence between both the states and the transitions of the LTS and those of such a stochastic process, analogous to the relation between the LTS of a BIO-PEPA system and the underlying CTMC.

A graphical representation of the state-transitions for the process is given in FIGURE 9.3. In that figure, all the states are represented as circles where the notation $(n_1, n_2) : m$ represents the discrete levels of concentration of the species A, n_1, and B, n_2. The number m represents the number of instances of the unique possible action α currently scheduled in the state. All the arrows represent stochastic derivations of the whole system, where the labels are exactly those computed by that relation. The full arrows represent stochastic derivations based on start derivation, empty arrows represent stochastic derivations based on completion derivation. For this particular example, any empty arrow built from a derivation with a rate r refers to the completion of the unique action started with the same rate r.

TABLE 9.7 presents a table showing the explicit mapping of the states described in FIGURE 9.3 and the corresponding process configuration obtained by the semantics in the LTS. As expected, this system, starting from the initial configuration, namely state $(3, 0) : 0$, eventually reaches the final state $(0, 3) : 0$, which corresponds to the final configuration

$$A(0, [\,]) \underset{\{\alpha\}}{\bowtie} B(3, [\,])$$

and to the vector $(0, 3)^T$, as it was for the corresponding BIO-PEPA system.

LTS state	Bio-PEPAd process configuration
$(3,0):0$	$A(3,[\,]) \underset{\{\alpha\}}{\bowtie} B(0,[\,])$
$(2,0):1$	$A(3,[(3,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(0,[(0,1,\alpha,\uparrow)])$
$(2,1):0$	$A(2,[\,]) \underset{\{\alpha\}}{\bowtie} B(1,[\,])$
$(1,0):2$	$A(1,[(3,1,\alpha,\downarrow),(2,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(0,[(0,1,\alpha,\uparrow),(0,1,\alpha,\uparrow)])$
$(1,1):1$	$A(1,[(2,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(1,[(1,1,\alpha,\uparrow)])$
$(1,2):0$	$A(1,[\,]) \underset{\{\alpha\}}{\bowtie} B(2,[\,])$
$(0,0):3$	$A(1,[(3,1,\alpha,\downarrow),(2,1,\alpha,\downarrow),(1,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(0,[(0,1,\alpha,\uparrow),(0,1,\alpha,\uparrow),(0,1,\alpha,\uparrow)])$
$(0,1):2$	$A(1,[(2,1,\alpha,\downarrow),(1,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(1,[(1,1,\alpha,\uparrow),(1,1,\alpha,\uparrow)])$
$(0,2):1$	$A(0,[(1,1,\alpha,\downarrow)]) \underset{\{\alpha\}}{\bowtie} B(2,[(2,1,\alpha,\uparrow)])$
$(0,3):0$	$A(0,[\,]) \underset{\{\alpha\}}{\bowtie} B(3,[\,])$

Table 9.7: A table stating the correspondence between the states represented in Figure 9.3 and the process configurations obtained by the semantics.

9.2.3 Analysis techniques in Bio-PEPAd

In this section we present some analysis techniques for Bio-PEPAd systems analogous to those discussed for Bio-PEPA systems. Firstly, we present the automatic translation of a Bio-PEPAd system into a set of Delay Differential Equations. Secondly, we discuss the encoding of Bio-PEPAd processes in Generalized Semi-Markov processes. Thirdly, we discuss how to apply the DDa to compute the stochastic time-evolution of a Bio-PEPAd model.

Translation in Delay Differential Equations

Whenever phenomena presenting a delayed effect are described in deterministic models based on differential equations, we know that we move from ODEs to DDEs. Hence it is natural, in the context of Bio-PEPAd, to reason about the translation of a model into a set of DDEs, as Bio-PEPA systems are translated into sets of ODEs.

In order to define the encoding it is important to recall that we defined Bio-PEPAd in terms of Bio-PEPA. This means that, given a system specification $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P\rangle$ where P is a valid Bio-PEPA process, we just need to modify the algorithm defined in (Ciocchetta & Hillston, 2009) and previously informally discussed, to add the information provided by σ concerning the delays. We already said that a Bio-PEPA system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle$ is encoded in a set of ODEs by using the definition of the stoichiometry matrix $D = \{d_{i,j}\}$ associated with P, the kinetic law vector ν_{KL} and by associating the deterministic variables with the components.

We discuss the steps of the algorithm:

(1) As in Bio-PEPA, the stoichiometry matrix is $D \in \mathbb{N}^{n\times m}$ if the system contains n distinct species which can perform m actions . Here, we assume we enumerate the actions as $\alpha_1, \ldots, \alpha_m$ and the species in the system as $S_1, \ldots, S_n$. The entry $d_{i,j}$, representing the change in the levels induced by performing action α_j with species S_i, is defined as follows:

$$d_{i,j} = \begin{cases} -\kappa_{i,j} & \text{if } (\alpha_j, \kappa_{i,j})\downarrow \text{ is an action for species } S_i \\ +\kappa_{i,j} & \text{if } (\alpha_j, \kappa_{i,j})\uparrow \text{ is an action for species } S_i \\ 0 & \text{otherwise.} \end{cases}$$

(2) The definition of the Bio-PEPAd m-dimensional kinetic law vector ν_{KL} is different from Bio-PEPA. In this step, we build, instead of a set of ODEs, a set of DDEs. We formally

define the entries in the vector only for the well-known kinetic functions f_{MA}, f_{MM} and f_H, arbitrary functions must be appropriately encoded by the modeler. Here we denote with S_1 and S_2 either two species involved as reactants in a mass-action kinetic, or two species involved as an enzyme (S_1) and a substrate (S_2) in a Michaelis-Menten kinetic, or, in the case of Hill kinetic, the only reactant is from species S_1. With x_i we denote the deterministic variable representing species S_i. The entry ν_{KLi} of the vector is defined as follows:

$$\nu_{KLi} = \begin{cases} kx_1(t-\sigma(\alpha_i))x_2(t-\sigma(\alpha_i)) & \text{if } f_{\alpha_i} = f_{MA}(k) \\ \dfrac{vx_1(t-\sigma(\alpha_i))x_2(t-\sigma(\alpha_i))}{K + x_2(t-\sigma(\alpha_i))} & \text{if } f_{\alpha_i} = f_{MM}(v,K) \\ \dfrac{vx_1(t-\sigma(\alpha_i))^p}{K + x_1(t-\sigma(\alpha_i))^p} & \text{if } f_{\alpha_i} = f_H(v,K,p) \end{cases}$$

(3) As in BIO-PEPA, now we associate the variable x_i with each species component S_i and so define the n-dimensional vector $\overline{x}$.

The DDE system can be defined in the same way as the ODE one, namely as

$$\frac{d\overline{x}}{dt} = D \times \nu_{KL}$$

where $\overline{x}$ and D are the results of step (3) and (1) of the algorithm, respectively. The initial conditions are, however, different from the ones defined for ODEs. In particular, the DDEs, because of the delays, must be defined also in the interval $[t_0 - \sigma(\alpha); t_0]$ where α is the action with maximum delay.

It is not possible to define a universal initial condition for the DDEs systems as every possible configuration will affect the dynamics of the whole system. Sometimes the initial conditions of a species S are defined via a constant function $\phi_S(t)$ for $t \in [t_0 - \sigma(\alpha), t_0]$ such that $\phi_S(t) = hl_{S,0}$ where $l_{S,0}$ is the initial concentration level for S in the BIO-PEPAD model and h is the step size for the concentration levels. In general, we leave this part of the translation to the modeler who has to tune the initial conditions with respect to the specification of the target system.

Mapping a 2-reactions BIO-PEPAD model. Let us consider an extension of the simple model we previously presented. The model considered a single reaction $A \overset{k,\sigma'}{\longmapsto} B$, modeled by the initial process configuration $A(3,[\ ]) \underset{\{\alpha\}}{\bowtie} B(0,[\ ])$ of the processes $A \overset{def}{=} (\alpha,1)\downarrow A$, and $B \overset{def}{=} (\alpha,1)\uparrow B$. We extend the model by considering the transformation to be reversible, namely the pair of reactions $A \overset{k_1,\sigma_1}{\longmapsto} B$ and $B \overset{k_2,\sigma_2}{\longmapsto} A$.

This pair of reactions can be modeled by simply extending the single-reaction model. Indeed, we define the processes

$$A \overset{def}{=} (\alpha,1)\downarrow + (\beta,1)\uparrow$$
$$B \overset{def}{=} (\alpha,1)\uparrow + (\beta,1)\downarrow$$

and $A(3) \underset{\{\alpha,\beta\}}{\bowtie} B(0)$ where β models the new reaction.

Encoding this simple process is straightforward. Firstly, the definition of the stoichiometry matrix is

$$D = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

since on the columns we assume to have the reactions in order of appearance, and on the rows the species A and B. As expected, $d_{1,1} = -1$ reflects the transformation of one A and $d_{1,2} = 1$ the B which is, consequently, produced.

The second step defines the kinetic law vector. Since there are two reactions, the vector is 2-dimensional, and is defined as

$$\nu_{KL} = \begin{pmatrix} k_1 x_1(t-\sigma(\alpha)) \\ k_2 x_2(t-\sigma(\beta)) \end{pmatrix} = \begin{pmatrix} k_1 x_1(t-\sigma_1) \\ k_2 x_2(t-\sigma_2) \end{pmatrix}$$

The third step associates the names x_1 and x_2 with the species A and B, respectively. Finally, the DDE system $d\overline{x}/dt = D \times \nu_{KL}$ is the following:

$$\frac{dA}{dt} = -k_1 A(t-\sigma_1) + k_2 B(t-\sigma_2) \qquad \frac{dB}{dt} = k_1 A(t-\sigma_1) - k_2 B(t-\sigma_2)\,.$$

By defining some initial conditions in $[t_0 - max\{\sigma_1, \sigma_2\}, t_0]$ the system could be either analytically or numerically studied.

Bio-PEPAd processes as Generalized Semi-Markov Processes

We discussed GSMPs in SECTION 2.2. We need to adapt the definition of GSMPs for our purposes, namely to have a mapping from Bio-PEPAd configurations and states of a GSMP, and a mapping from Bio-PEPAd transitions and GSMP transitions.

The running activities of a GSMP correspond to Bio-PEPAd actions which are started and not completed. In this sense, by considering the definition of Bio-PEPAd, we can consider two types of activities, those modeling the start of a new action, and those modeling the completion of an action. By the definition of GSMP we know that to each *active elements* is associated a *lifetime*. In this sense, each enabled action in the starting relation correspond to exponentially distributed active elements of the state, as it is for an action starting in the DDA, it consumes an exponentially distributed quantity of time. In contrast, those actions which are currently progressing through a delay, namely those currently contained in the scheduling lists of the components, correspond to the generally distributed active elements of state. Of course, those activities are associated to deterministic constant distributions.

The definition of a GSMP can be rephrased as follows. The set of states $\{x \mid x \in X\}$ is left unchanged, and for each x there are active elements $s \in S$. Accordingly to the intuition we discussed, the set of active elements S can be partitioned into two sets S' and $S^\star$ such that $S' \cap S^\star = \emptyset$. The former set collects elements with exponentially distributed lifetime, and the latter elements with deterministic constant lifetime. As expected, elements decay at rate $r(s,x)$ and when an active element s dies the process moves to state $x' \in X$ with probability $p(x,s,x')$. When a transition is generated by the starting relation it corresponds to the death of an exponentially distributed active element. When a transition is generated by the completion relation it corresponds to the death of a deterministically distributed active element. We recall that all interrupted elements record their residual lifetimes, and whenever the element is again active it resumes its remaining lifetime. Also, if the lifetimes are exponential we may disregard the residual lifetimes, restarting each element with a new lifetime whenever it is active. Notice that this corresponds to the DDA, as expected.

Finally, we remark that by the considerations we did on timing aspects in the Bio-PEPAd semantics there could be transitions in the LTS which should be assigned probability 0 in the GSMP. We discuss why this is guaranteed by the definition of GSMPs. The fact that when jumping over the states of a GSMP the process records and resumes residual lifetimes of events, makes impossible to jump trough transitions which effectively can not happen. This impossibility is guaranteed by the definition of a GSMP, and makes practically these transitions to have probability 0, which is in fact what we required.

Stochastic Simulation of Bio-PEPAd systems

The stochastic simulation of Bio-PEPAd systems is to be performed by the DDA since delays of the actions are assumed to follow such an approach. However, the techniques we present to encode

a Bio-PEPAd system are general and independent on the target algorithm. In this sense, we are able to encode Bio-PEPAd systems in input valid also for PDa simulations.

As above the Bio-PEPA approach forms the basis for Bio-PEPAd analysis. In particular, we are able to re-use parts of the method defined in (Ciocchetta & Hillston, 2009) to perform stochastic simulation of Bio-PEPA systems using the SSa.

The main steps in preparing a Bio-PEPAd system for the application of the DSSa are two: define the algebraic representation of a process, and create the reactions to simulate. We introduce a family of functions

$$\left\{ (\!|\_|\!)_n : \mathcal{C} \cup \mathcal{P} \mapsto \mathbb{N}^n \,|\, n \in \mathbb{N},\ n \geq 0 \right\}$$

for the encoding of either a Bio-PEPA process or a Bio-PEPAd process configuration in a n-dimensional vector such that:

$$(\!|S_1(l_1) \underset{\mathcal{L}_1}{\bowtie} \ldots \underset{\mathcal{L}_m}{\bowtie} S_m(l_m)|\!)_m = (l_1, \ldots, l_m)^T$$
$$(\!|S_1(l_1, L_1) \underset{\mathcal{L}_1}{\bowtie} \ldots \underset{\mathcal{L}_n}{\bowtie} S_n(l_n, L_n)|\!)_n = (l_1, \ldots, l_n)^T .$$

Since we assume we have a well-defined Bio-PEPA model all the species S_i are distinct, and are represented by a distinct element in the resulting vector. In the following, we use $(\!|\_|\!)$ to denote the function mapping a Bio-PEPA process or a Bio-PEPAd process configuration to an appropriate vector-space. For instance, for the example we previously discussed, the encoding is such that $(\!|A(3,[\,]) \underset{\{\alpha\}}{\bowtie} B(0,[\,])|\!) = (3,0)^T$, as required. Given an initial system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P \rangle$ a vector describing the initial number of molecules to be simulated is defined, given $|Comp| = n$, as

$$\mathbf{x}_0 = \begin{pmatrix} (\!|P|\!)_1 \\ \ldots \\ (\!|P|\!)_n \end{pmatrix} HN_A v$$

where H is the step size of the system, v is the volume of the target compartment and N_A is the Avogadro number.

Secondly, the actual rates of the reactions have to be defined, case by case, using the parameters in $\mathcal{F}$. Since $\mathcal{F}$ is defined in the same way for both Bio-PEPA and Bio-PEPAd, the techniques developed in (Ciocchetta & Hillston, 2009) to derive rates from $\mathcal{F}$ the actual rates can be also applied in the context of Bio-PEPAd. Once these two steps have been performed, the resulting system can be simulated by the DDa.

Mapping the 2-reactions Bio-PEPAd model. Let us consider the same Bio-PEPAd model translated in DDEs in the previous section. By applying the techniques we introduced the processes $A \overset{def}{=} (\alpha,1)\downarrow + (\beta,1)\uparrow$, $B \overset{def}{=} (\alpha,1)\uparrow + (\beta,1)\downarrow$ and $A(3) \underset{\{\alpha,\beta\}}{\bowtie} B(0)$ can be encoded in the 2-reactions model

$$R_1 :\ A \xrightarrow{k_1,\sigma_1} B \qquad\qquad R_2 :\ B \xrightarrow{k_2,\sigma_2} A$$

and the initial state vector can be defined as

$$\mathbf{X}(t_0) = \begin{pmatrix} n_0^A HN_A v \\ n_0^B hN_A v \end{pmatrix} = \begin{pmatrix} 3HN_A v \\ 0 \end{pmatrix} .$$

Finally, t_0, $\mathbf{X}(t_0)$ and $\{R_1, R_2\}$, together with the information about the propensity functions for the reactions, can be used as input for the DDa to analyze the model.

9.3 Relation between Bio-PEPAd and Bio-PEPA

In this section we prove theorems stating the correspondence between the semantics of Bio-PEPA and Bio-PEPAd. More precisely, we start by introducing a notion of *interchangeability* between

Bio-PEPA processes and Bio-PEPAd process configurations. Through a series of results on interchangeable processes we show that the LTS of a Bio-PEPA process can be obtained from the LTS of the corresponding interchangeable Bio-PEPAd process configuration. Moreover we relate probabilities in the LTS of such a process configuration with those in the LTS of the Bio-PEPA process.

We start by defining the inverse of function μ, denoted as $\mu^{-1} : \mathcal{C} \mapsto \mathcal{P}$ and used to transform a Bio-PEPAd process configuration in a Bio-PEPA process

$$\mu^{-1}((\alpha,\kappa)op\ S) = (\alpha,\kappa)op\ S \qquad \mu^{-1}(P_1 \underset{\mathcal{L}}{\bowtie} P_2) = \mu^{-1}(P_1) \underset{\mathcal{L}}{\bowtie} \mu^{-1}(P_2)$$
$$\mu^{-1}(S_1 + S_2) = S_1 + S_2 \qquad \mu^{-1}(S(l,L)) = S(l)\,.$$

As reasonably expected function μ is not bijective, namely even if in μ a unique Bio-PEPAd process configuration corresponds to each Bio-PEPA process, the opposite is generally false. Indeed it is the case that $\forall L \in \mathcal{L}_{\mathcal{D}}.\mu^{-1}(S(l,L)) = S(l)$, which means that we lose information about the structure of L, namely the actions started and not yet completed in $S(l,L)$.

We want to concentrate on those processes for which it is reasonable to define a valid notion of interchangeability.

Definition (Interchangeable processes) A Bio-PEPA process $P \in \mathcal{P}$ and a Bio-PEPAd process configuration $P_C \in \mathcal{C}$ are said to be *interchangeable* if and only if

$$\mu(P) = P_C \ \wedge\ \mu^{-1}(P_C) = P\,.$$

Note that if P and P_C are interchangeable, then by definition $\mu(P) = P_C$ and, consequently, all the lists appearing in P_C must be empty. Practically, in P_C there must be no uncompleted actions running. Intuitively, this definition is constrained by the structure of Bio-PEPA processes which cannot have concurrently running uncompleted actions. Alternatively we could have defined μ^{-1} only on empty lists, i.e. as $\mu^{-1}(S(l,[\,])) = S(l)$, but in this case μ^{-1} would not have been a total function.

Now we prove the following theorem on the relation of interchangeability.

Theorem 9.3.1. *Let* $\mathcal{I} = \{(P, P_C) \mid P \in \mathcal{P}, P_C \in \mathcal{C}, \mu(P) = P_C, \mu^{-1}(P_C) = P\}$, *then*

$$\forall (P,P_C) \in \mathcal{I}.\ \forall P' \in \mathcal{P}.P \xrightarrow{(\alpha,w)}_c P' \implies$$
$$\exists P'_C, P''_C \in \mathcal{C}.P_C \xrightarrow{(\alpha^+,w)}_{st} P'_C \xrightarrow{(\alpha^-,w)}_{co} P''_C \ \wedge\ (P', P''_C) \in \mathcal{I}$$

Proof. We prove the theorem inductively on the structure of P. Before presenting the proof by cases we state the following equalities which hold for any (l,κ,α,op):

$$\phi\,\alpha\,[(l,\kappa,\alpha,op)] = (l,\kappa,\alpha,op) \qquad \zeta\,\alpha\,[(l,\kappa,\alpha,op)] = [\,]\,. \tag{9.1}$$

- Let us consider $P \equiv ((\alpha,\kappa)\downarrow S)(l)$ which is interchangeable with $P_C \equiv ((\alpha,\kappa)\downarrow S)(l,[\,])$; from P there exist a unique possible derivation $P \xrightarrow{(\alpha,[S:\downarrow(l,\kappa)])}_s S(l-\kappa) \equiv P'$ if $k \le l \le N$. With the same condition from P_C we can derive $P_C \xrightarrow{(\alpha^+,[S:\downarrow(l,\kappa)])}_{st} S(l-\kappa,[(l,\kappa,\alpha,\downarrow)]) \equiv P'_C$; among all the possible derivations from P'_C there is one such that $P'_C \xrightarrow{(\alpha^-,[S:\downarrow(l,\kappa)])}_{co} S(l-\kappa,[\,]) \equiv P''_C$ by using equation (9.1). Finally, $(P', P''_C) \in \mathcal{I}$ since $\mu(S(l-\kappa)) = S(l-\kappa,[\,]) \equiv P''_C$.
- Let us consider $P \equiv ((\alpha,\kappa)\uparrow S)(l)$ and $P_C \equiv ((\alpha,\kappa)\uparrow S)(l,[\,])$; by similar arguments for the previous case we have $P \xrightarrow{(\alpha,[S:\uparrow(l,\kappa)])}_s (\alpha,\kappa)\uparrow S)(l+\kappa)$ if $0 \le l \le N-\kappa$, we have also $P_C \xrightarrow{(\alpha^+,[S:\uparrow(l,\kappa)])}_{st} S(l,[(l,\kappa,\alpha,\uparrow)]) \equiv P'_C$ and $P'_C \xrightarrow{(\alpha^-,[S:\uparrow(l,\kappa)])}_{co} S(l+\kappa,[\,]) \equiv P''_C$ by using equation (9.1). Finally, $(P', P''_C) \in \mathcal{I}$ since $\mu(S(l+\kappa)) = S(l+\kappa,[\,]) \equiv P''_C$.

- Let us consider $P \equiv ((\alpha,\kappa)opS)(l)$ and $P_C \equiv ((\alpha,\kappa)opS)(l,[\,])$ with $op = \oplus, \ominus, \odot$; by similar arguments for the previous case we have $P \xrightarrow{(\alpha,[S:op(l,\kappa)])}_s (\alpha,\kappa)opS)(l)$ with the proper condition on levels depending on op; we have also $P_C \xrightarrow{(\alpha^+,[S:op(l,\kappa)])}_{st} S(l,[(l,\kappa,\alpha,op)]) \equiv P'_C$ and $P'_C \xrightarrow{(\alpha^-,[S:op(l,\kappa)])}_{co} S(l,[\,]) \equiv P''_C$ by using equation (9.1). Finally, $(P', P''_C) \in \mathcal{I}$ since $\mu(S(l)) = S(l,[\,]) \equiv P''_C$.

- Let us consider $P \equiv (S_1 + S_2)(l)$ and $P_C \equiv (S_1 + S_2)(l,[\,])$; we have that $P \xrightarrow{\alpha,w}_s S'_1(l') \equiv P'$ if $S_1(l) \xrightarrow{\alpha,w}_s S'_1(l')$. We assume the inductive hypothesis on $S_1(l)$ which means that $(S_1(l), P_{S_1}(l,[\,])) \in \mathcal{I}$, $S_1(l) \xrightarrow{(\alpha,w)}_s S'_1(l')$ and $(S'_1(l'), P_{S'_1}(l',[\,])) \in \mathcal{I}$. By considering Bio-PEPAd semantics we have that $P_C \xrightarrow{(\alpha^+,w)}_{st} S'_1(l',L) \equiv P'_C \xrightarrow{(\alpha^-,w)}_{co} S'_1(l',[\,])$ if $S_1(l,[\,]) \xrightarrow{(\alpha^+,w)}_{st} S'_1(l',L)$ and $S'_1(l',L) \xrightarrow{(\alpha^-,w)}_{co} S'_1(l',[\,])$. By applying the inductive hypothesis and by considering that L must contain only one element so we can apply equation (9.1), we have that $P_{S_1}(l,[\,]) \equiv S_1(l,[\,])$,$P'_C = S'_1(l',L)$ and $P''_C \equiv S'_1(l',[\,]) \equiv P_{S'_1}(l')$. The case in which is S_2 to move is symmetric.

- The case for $P \equiv C(l)$ comes from the inductive hypothesis on $S(l)$ once that $C \stackrel{def}{=} S$; namely $P' \equiv S'(l')$, $P_C \equiv S(l,[\,])$ and $P''_C \equiv S'(l',[\,])$.

- Let us consider $P \equiv P_1 \underset{\mathcal{L}}{\boxtimes} P_2 \xrightarrow{\alpha,w} P'_1 \underset{\mathcal{L}}{\boxtimes} P_2$ if $\alpha \notin \mathcal{L}$ and $P_1 \xrightarrow{\alpha,w} P'_1$. Let us assume the inductive hypothesis on P_1; namely $\exists P_{P_1}, P_{P'_1}$ such that $(P_1, P_{P_1}), (P'_1, P_{P'_1}) \in \mathcal{I}$. Let us consider $P_C \equiv \mu(P_1 \underset{\mathcal{L}}{\boxtimes} P_2) = \mu(P_1) \underset{\mathcal{L}}{\boxtimes} \mu(P_2)$; we have that $P_C \xrightarrow{(\alpha^+,w)}_{st} P'_{1,\mu} \underset{\mathcal{L}}{\boxtimes} \mu(P_2) \equiv P'_C$ if $\alpha \notin \mathcal{L}$ and $\mu(P_1) \xrightarrow{(\alpha^+,w)}_{st} P'_{1,\mu}$. Notice that since in $\mu(P_1)$ no actions are running, the same is not true in $P'_{1,\mu}$, since α just started. Also, we have that $P'_{1,\mu} \underset{\mathcal{L}}{\boxtimes} \mu(P_2) \xrightarrow{(\alpha^-,w)}_{co} \mu(P'_1) \underset{\mathcal{L}}{\boxtimes} \mu(P_2)$ if $P'_{1,\mu} \xrightarrow{(\alpha^-,w)}_{co} \mu(P'_1)$; by the inductive hypothesis $P_{P_1} \equiv \mu(P_1)$, $P_{P'_1} \equiv \mu(P'_1)$ and consequently $P''_C \equiv \mu(P'_1) \underset{\mathcal{L}}{\boxtimes} \mu(P_2) = \mu(P'_1 \underset{\mathcal{L}}{\boxtimes} P_2)$ which is interchangeable with $P'_1 \underset{\mathcal{L}}{\boxtimes} P_2$. The case in which is P_2 to move is symmetric. Also, the case in which both P_1 and P_2 move is a generalization of this case with two inductive hypothesis.

□

This theorem states a constructive result for the interchangeable relation stating that if P and P_C are interchangeable, then for any possible action derivable from P and leading to a state P', there exists a sequence of start and completion transitions, from P_C to P''_C through P'_C, such that P' and P''_C are interchangeable. Thus we can think of interchangeability as a *simulation*, since everything which can be done from a Bio-PEPA process can be done by the interchangeable Bio-PEPAd process configuration. However note that P'_C is not interchangeable to P' since $\mu(P') \neq P'_C$ and it is fairly easy to see that $\nexists P'' \in \mathcal{P}.P''$ and P'_C are interchangeable. This confirms our intuition that interchangeability is not a *bisimulation* as it is not symmetric. In particular,

$$\exists (P,P_C) \in \mathcal{I}.\ \exists P'_C, P''_C \in \mathcal{C}.$$
$$P_C \xrightarrow{(\alpha^+,w)}_{st} P'_C \xrightarrow{(\alpha^-,w)}_{st} P''_C \ \wedge\ \nexists P'' \in \mathcal{P}.(P'', P''_C) \in \mathcal{I}$$

Let us denote a generic Bio-PEPA system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle$ as $\langle P\rangle$ whenever we are not concerned with the components of the system itself, and let us do the same for Bio-PEPAd system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P\rangle$ by writing $\langle \sigma, P\rangle$; we can now prove the following theorem.

Theorem 9.3.2. *For any system* $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle$ $\exists P_C \in \mathcal{C}.(P,P_C) \in \mathcal{I}$ *such that*

$$\forall P' \in \mathcal{P}.\ \langle P\rangle \xrightarrow{(\alpha,r)}_s \langle P'\rangle \implies \forall \sigma \in \Delta.\langle \sigma, P_C\rangle \xrightarrow{(\alpha^+,r)}_s \langle \sigma, P'_C\rangle \wedge$$
$$\langle \sigma, P'_C\rangle \xrightarrow{(\alpha^-,r)}_s \langle \sigma, P''_C\rangle \ \wedge\ (P', P''_C) \in \mathcal{I}$$

Proof. The proof is done by induction on the structure of the systems with a schema similar to proof of THEOREM (9.3.1); the inductive hypothesis assumed on capability and completion relations are results of THEOREM (9.3.1). □

This theorem extends the notion of interchangeability to systems in a natural way. More precisely, if two processes are interchangeable, then any of the possible BIO-PEPA systems is interchangeable to an infinity of different BIO-PEPAD systems.

After these general results on interchangeability we can easily notice that there always exists a configuration interchangeable to any BIO-PEPA process, moreover such a configuration is unique. More formally, each process is interchangeable uniquely with the process configuration obtained by applying μ, namely $\forall P \in \mathcal{P}.(P, \mu(P)) \in \mathcal{I}$, and this is easily verifiable since by definition $\mu^{-1}(\mu(P)) = P$.

This means that we can build the BIO-PEPA stochastic semantics of a process P, namely its LTS, by considering the semantics of a generic BIO-PEPAD system starting in $\mu(P)$ and traversing only interchangeable configurations. We state this theorem which formally characterizes the BIO-PEPA stochastic relation by means of BIO-PEPAD stochastic relation.

COROLLARY 9.3.3. *The* BIO-PEPA *stochastic relation $\rightarrow_s$ is equivalently defined by the following inference rule*

$$\frac{\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, \mu(P)\rangle \xrightarrow{(\alpha^+, r_\alpha[w,\mathcal{N},\mathcal{K}], \sigma(\alpha))}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P'_C\rangle \quad \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, P'_C\rangle \xrightarrow{(\alpha^-, r_\alpha[w,\mathcal{N},\mathcal{K}], \sigma(\alpha))}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, \mu(P')\rangle}{\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle \xrightarrow{(\alpha, r_\alpha[w,\mathcal{N},\mathcal{K}])}_s \langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P'\rangle}$$

where σ is a generic function from Δ.

Proof. The proof comes from noting that THEOREM 9.3.2 states a strong relationship between the stochastic derivations of BIO-PEPA processes and the corresponding stochastic derivations of BIO-PEPAD process configurations. More precisely, if we rephrase THEOREM (9.3.2) considering that $(P, \mu(P)) \in \mathcal{I}$ we have that, for any BIO-PEPA system $\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, P\rangle$ there exists a BIO-PEPAD process configuration $\mu(P)$ such that P and $\mu(P)$ are interchangeable and any stochastic derivation from $\langle P\rangle$ to $\langle P'\rangle$ for action α and rate r, can be equivalently described by two stochastic derivations from $\langle \sigma, \mu(P)\rangle$ to $\langle \sigma, \mu(P')\rangle$ through a configuration $\langle \sigma, P'_C\rangle$ for the same action α and the same rate r. □

We can apply these results and definitions to the toy examples discussed earlier. In particular, we have the interchangeability described by the following set

$$\begin{aligned}\mathcal{I} = \{&(A(3) \underset{\{\alpha\}}{\bowtie} B(0), A(3,[\,]) \underset{\{\alpha\}}{\bowtie} B(0,[\,])), (A(2) \underset{\{\alpha\}}{\bowtie} B(1), A(2,[\,]) \underset{\{\alpha\}}{\bowtie} B(1,[\,]))\\ &(A(1) \underset{\{\alpha\}}{\bowtie} B(2), A(1,[\,]) \underset{\{\alpha\}}{\bowtie} B(2,[\,])), (A(0) \underset{\{\alpha\}}{\bowtie} B(3), A(0,[\,]) \underset{\{\alpha\}}{\bowtie} B(3,[\,]))\}\,.\end{aligned}$$

As a consequence, the LTS of the BIO-PEPA process shown in FIGURE 9.1 can be obtained by applying results of THEOREM 9.3.3.

The theorem we proved relates BIO-PEPA semantics and BIO-PEPAD semantics. A final point relating to probabilities is worth discussing. We know that BIO-PEPAD models can be simulated by the DDA or by analysis techniques based on GSMPs. Thus we might ask, for instance, what is the probability of observing in a DDA simulation a sequence of configuration changes such that the configurations are interchangeable to some processes? More precisely, given $P_C = \mu(P)$ and $P''_C = \mu(P')$ we aim to derive an analytical formula for the probability of the stochastic derivations

$$\forall \alpha \in \mathcal{A}.\ \langle \sigma, \mu(P)\rangle \xrightarrow{(\alpha^+, r_\alpha[w,\mathcal{N},\mathcal{K}])}_s \langle \sigma, P'_C\rangle \xrightarrow{(\alpha^-, r_\alpha[w,\mathcal{N},\mathcal{K}])}_s \langle \sigma, \mu(P')\rangle\,.$$

As we know, for each configuration there is a corresponding vector in the state space, so we have that

$$\begin{aligned}(\!|\mu(P)|\!) &= \mathbf{x}\\ (\!|P'_C|\!) &= \mathbf{x} + \nu^r_\alpha\\ (\!|\mu(P')|\!) &= \mathbf{x} + \nu^r_\alpha + \nu^p_\alpha = \mathbf{x} + \nu_\alpha\end{aligned}$$

where ν^r_α and ν^p_α denote the stoichiometry vector for the reactants and the products and are such that $\nu_\alpha = \nu^r_\alpha + \nu^p_\alpha$. We can now state the following proposition.

Proposition 9.3.4. *The probability $p(\mathbf{x})$ of observing equivalent state changes (i.e. $\mathbf{x}$ modified in $\mathbf{x} + \nu^r_\alpha$ modified in $\mathbf{x} + \nu_\alpha$) is given in the* DDA *by the quantity*

$$p(\mathbf{x}) = \sum_{i=1}^{M} \frac{a_i(\mathbf{x})}{a_0(\mathbf{x})} e^{-a_0(\mathbf{x}+\nu_i^R)\sigma(i)}\,. \tag{9.2}$$

Such a formula is derived accordingly to the following arguments. When the system is in state $\mathbf{x}$ at time t, the next value for $\tau \sim Exp(a_0(\mathbf{x}))$ is sampled and a reaction is chosen to fire with probability $a_j(\mathbf{x})/a_0(\mathbf{x})$; notice that no reactions are scheduled in the system since $P_C \equiv \mu(P)$. Assuming to chose reaction α, state is changed from $\mathbf{x}$ to $\mathbf{x} + \nu^r_\alpha$ and time is increased to $t + \tau$. At the next step, a new value for $\tau' \sim Exp(a_0(\mathbf{x} + \nu^r_\alpha))$ is sampled: if $\tau' > \sigma(\alpha)$ then the state changes to $\mathbf{x} + \nu_\alpha$ and time to $t + \sigma(\alpha)$, otherwise a new reaction is scheduled. Our target event is $\tau' > \sigma(\alpha)$ which has probability $\exp(-a_0(\mathbf{x} + \nu^r_\alpha)\sigma(\alpha))$. Since events are independent, if we generalize among all possible reactions we get equation (9.2).

If we consider equation (9.2) in systems where $\forall\alpha.\ f_\alpha[w, \mathcal{N}, \mathcal{K}] = a_\alpha(\mathbf{x})$, by considering the Bio-PEPA definitions of *α-derivative* and *exit rate* for a process (Ciocchetta & Hillston, 2009; Ciocchetta & Hillston, 2008) rephrased in Bio-PEPAd, we can state the following proposition.

Proposition 9.3.5. *In* Bio-PEPAd*, the probability logically equivalent to $p(\mathbf{x})$ for $\mu(P)$ is given by equation*

$$\mathbb{P}(\mu(P)) = \sum_{i=1}^{M} \frac{f_i[w, \mathcal{N}, \mathcal{K}]}{ExitRate(\mu(P))} e^{-ExitRate(\mu(P))\sigma(i)}\,. \tag{9.3}$$

This is an interesting result relating Bio-PEPAd and Bio-PEPA probabilities in the stochastic regime since

$$\lim_{\sigma\to\infty} \mathbb{P}(\mu(P)) = 0 \qquad\qquad \lim_{\sigma\to 0} \mathbb{P}(\mu(P)) = \sum_{i=1}^{M} \frac{f_i[w, \mathcal{N}, \mathcal{K}]}{ExitRate(\mu(P))}$$

where $\sigma \to k$ means $\forall i = 1, \ldots, M.\ \sigma(i) \to k$. In particular, in the limit $\sigma \to 0$ equation (9.3) reduces to the probability of all the outgoing transitions from P, in its associated Ctmc.

The probability of all the possible paths which satisfy the interchangeability property is given as the recursive closure of $\mathbb{P}(\mu(P))$. This is the probability of observing, during a simulation of a Bio-PEPAd model, a series of steps which correspond to the interchangeable Bio-PEPA process. So, for instance, for the toy example, the probability of observing the sequence of state changes

$$(3,0) : 0 \to (2,0) : 1 \to (2,1) : 0 \to (1,1) : 1 \to (1,2) : 0 \to (0,2) : 1 \to (0,3) : 0$$

which is conceptually equivalent to the non-delayed sequence

$$(3,0) : 0 \ \to (2,1) : 0 \to (1,2) : 0 \to (0,3) : 0$$

is given by $\mathbb{P}(\tau > \sigma';\ \tau' > \sigma')$ which, since $\tau \sim Exp(2k)$ and $\tau' \sim Exp(k)$ are independent, evaluates as $e^{-3k\sigma'}$. Note that here the unique reaction is chosen with probability 1.

9.4 Examples Bio-PEPAd models

In the following section we encode the target models presented in Section 3.1.1-3.1.3 and the cell cycle model presented in Section 3.2 in Bio-PEPAd.

Cellular Models

Let us consider the protein transcription/translation model presented in Section 3.1.1, in this section we discuss how such model can be represented in Bio-PEPAd, and some

As in either the deterministic and stochastic definition of the model, we assume four Bio-PEPAd species, one for each possible molecule: the DNA, DNA, the mRNA, $mRNA$, the complex, $DNA{:}P$, and the protein, P. We model transcription and translation with actions α_1 and α_2, respectively, the binding and the unbinding with actions α_3 and α_4, respectively. Moreover, degradation events are modeled with actions α_5 and α_6, respectively.

The Bio-PEPAd processes are defined as follows

$$\begin{aligned} DNA &\stackrel{def}{=} (\alpha_1,1) \oplus DNA + (\alpha_3,1)\downarrow DNA + (\alpha_4,1)\uparrow DNA \\ mRNA &\stackrel{def}{=} (\alpha_1,1)\uparrow mRNA + (\alpha_2,1) \oplus mRNA + (\alpha_5,1)\downarrow mRNA \\ P &\stackrel{def}{=} (\alpha_2,1)\uparrow P + (\alpha_6,1)\downarrow P \\ DNA:P &\stackrel{def}{=} (\alpha_3,1)\uparrow DNA:P + (\alpha_4,1)\downarrow DNA:P\,. \end{aligned}$$

Notice that the DNA and $mRNA$, which are involved in actions α_1 and α_2 modeling transcription and translation, since they are not consumed, they take part in the action as activators. In this sense, this guarantees from the point of view of the kinetic of the reaction, they the contribution of their concentration level is considered, as we expect. The whole cooperating process is defined as

$$DNA(l_D) \underset{\mathcal{L}_1}{\bowtie} mRNA(l_R) \underset{\mathcal{L}_2}{\bowtie} P(l_P) \underset{\mathcal{L}_3}{\bowtie} DNA:P(l_{D_P})$$

where $\mathcal{L}_1 = \{\alpha_1, \alpha_3, \alpha_4\}$, $\mathcal{L}_2 = \{\alpha_2, \alpha_3, \alpha_4\}$ and $\mathcal{L}_3 = \{\alpha_3, \alpha_4\}$. As expected, the actions in the cooperations set are only the binary ones and l_D, l_R, I_P and l_{D_P} denote the initial concentration levels for the species. According to the kinetics underlying the model the rate functions can be defined as

$$\begin{aligned} f_{\alpha_1} &= f_{MA}(\alpha) & f_{\alpha_2} &= f_{MA}(\beta) \\ f_{\alpha_3} &= f_{MA}(\gamma_1) & f_{\alpha_4} &= f_{MA}(\gamma_2) \\ f_{\alpha_5} &= f_{MA}(\delta_1) & f_{\alpha_6} &= f_{MA}(\delta_2)\,. \end{aligned}$$

Delay specification for this model are straightforward since the function σ is such that

$$\sigma(\alpha_1) = \sigma_1 \qquad\qquad \sigma(\alpha_2) = \sigma_2$$

and $\sigma(\alpha_i) = 0$ for $i = 1, \ldots, 4$. Here σ_1 and σ_2 represent the delay for transcription and translation. Given σ the system

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, DNA(l_D, [\,]) \underset{\mathcal{L}_1}{\bowtie} mRNA(l_R, [\,]) \underset{\mathcal{L}_2}{\bowtie} P(l_P, [\,]) \underset{\mathcal{L}_3}{\bowtie} DNA:P(l_{D_P}, [\,]) \rangle$$

can be analyzed by using the techniques discussed in the previous sections. More precisely, the LTS of the system can be generated once the original Bio-PEPA semantics has been combined with the one with delays (i.e. non-delayed actions must fire in a single step). Moreover, it it is fairly easy to notice that the DDEs and the DDA input which can be obtained as an encoding of this model by using the techniques we developed correspond to the one we discussed in Section 3.1.1.

Epidemic Models

Let us consider the SIR epidemic model presented in SECTION 3.1.2, in this section we discuss how such model can be represented in BIO-PEPAD.

As in either the deterministic and stochastic definition of the model, we assume three BIO-PEPAD species, one for each population: susceptible, S, infected, I and recovered, R. We model the renewal of the susceptible population with action α_1, the death of susceptible, infected and recovered with actions α_2, α_3 and α_4, respectively. Moreover, infection and recovery events are modeled with actions α_5 and α_6, respectively.

The BIO-PEPAD processes are defined as follows

$$\begin{aligned} S &\stackrel{def}{=} (\alpha_1, 1)\uparrow S + (\alpha_2, 1)\downarrow S + (\alpha_5, 1)\downarrow S \\ I &\stackrel{def}{=} (\alpha_3, 1)\downarrow I + (\alpha_5, 1)\uparrow I + (\alpha_6, 1)\downarrow I \\ R &\stackrel{def}{=} (\alpha_4, 1)\downarrow R + (\alpha_6, 1)\uparrow R \end{aligned}$$

and the whole cooperating process is defined as

$$S(l_S) \underset{\{\alpha_5\}}{\bowtie} I(l_I) \underset{\{\alpha_6\}}{\bowtie} R(l_R)$$

where, as expected, the actions in the cooperations set are only the binary ones and l_S, l_I and l_R denote the initial concentration levels for the species. According to the kinetics underlying the model the rate functions can be defined as

$$\begin{aligned} f_{\alpha_1} &= \alpha & f_{\alpha_2} &= f_{MA}(\delta_1) \\ f_{\alpha_3} &= f_{MA}(\delta_1) & f_{\alpha_4} &= f_{MA}(\delta_2) \\ f_{\alpha_5} &= \beta[I][S] & f_{\alpha_6} &= f_{MA}(\gamma)\,. \end{aligned}$$

Notice that functional rate for α_5 is defined in explicit mass-action style for the corresponding reaction $S + I \xrightarrow{\beta,\sigma_1} 2I$; this is done since the definition of well-formed BIO-PEPA process requires a process to have a unique term for each action. Hence we did not define the process as $(\alpha_5, 1)\downarrow I + (\alpha_5, 2)\uparrow I$ and the functional rate of α_5 as $f_{\alpha_5} = f_{MA}(\beta)$. Delay specification for this model are given by function σ

$$\sigma(\alpha_5) = \sigma_1 \qquad\qquad \sigma(\alpha_6) = \sigma_2$$

and $\sigma(\alpha_i) = 0$ for $i = 1, \ldots, 4$. Given σ the system

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, S(l_S, [\,]) \underset{\{\alpha_5\}}{\bowtie} I(l_I, [\,]) \underset{\{\alpha_6\}}{\bowtie} R(l_R, [\,]) \rangle$$

can be analyzed by using the techniques discussed in the previous sections. The same considerations of the previously presented BIO-PEPAD system applies here where the DDEs and the DDA input which can be obtained as an encoding of this model correspond to the one we discussed in SECTION 3.1.2.

Evolutionary Models

Let us consider the prey-predator model presented in SECTION 3.1.3, in this section we discuss how such model can be represented in BIO-PEPAD.

As in either the deterministic and stochastic definition of the model, we assume two BIO-PEPAD species, one for each population: the preys, X, and the predators, Y. We model the growth of the preys population with action α_1, the death of preys and predators with actions α_2 and α_3, respectively. Finally, the delayed predation is modeled with action α_4.

The BIO-PEPAD processes are defined as follows

$$X \stackrel{def}{=} (\alpha_1, 1)\uparrow X + (\alpha_2, 1)\downarrow X + (\alpha_4, 1)\downarrow X$$
$$Y \stackrel{def}{=} (\alpha_3, 1)\downarrow Y + (\alpha_4, 1)\uparrow Y\,.$$

and the whole cooperating process is defined as

$$X(l_X) \underset{\{\alpha_4\}}{\bowtie} Y(l_Y)$$

where, as expected, the actions in the cooperations set are only the binary ones and l_X and l_Y denote the initial concentration levels for the species. According to the kinetics underlying the model the rate functions can be defined as

$$f_{\alpha_1} = \alpha[X] \qquad f_{\alpha_2} = \alpha\gamma^{-1}[X]^2$$
$$f_{\alpha_3} = f_{MA}(\delta) \qquad f_{\alpha_4} = f_{MA}(\beta)\,.$$

Notice that, similarly to the previous model we presented, the preys perform α_1 with rate function f_{α_1} and α_2 with rate function f_{α_2} by the same considerations we outlined for α_5 in the BIO-PEPAD version of the SIR model.

Delay specification for this model are again straightforward since the function σ is such that

$$\sigma(\alpha_4) = \overline{\sigma} \qquad \sigma(\alpha_i) = 0 \quad i = 1, \ldots, 3\,.$$

Here $\overline{\sigma}$ represents the delay for predation. Given σ the system

$$\langle \mathcal{V}, \mathcal{N}, \mathcal{K}, \mathcal{F}, Comp, \sigma, X(l_X, [\,]) \underset{\{\alpha_4\}}{\bowtie} Y(l_Y, [\,])\rangle$$

can be analyzed by using the techniques discussed in the previous sections. The same considerations of the previously presented BIO-PEPAD system applies here where the DDEs and the DDA input which can be obtained as an encoding of this model correspond to the one we discussed in SECTION 3.1.3.

9.4.1 A model of the cell cycle with delays

In this section we encode in BIO-PEPAD the model of the cell cycle with delays we presented in SECTION 3.2.

The BIO-PEPAD model considers two populations of cells: T_I, the population of tumor cells during cell cycle interphase, and T_M, the population of tumor cells during mitosis. We consider four possible actions, α, β, γ and δ, one for each of the events that we want to model. In particular, action α models the passage from the interphase to the mitotic phase, with rate a_1, β models the mitosis, with rate a_4, γ the death of a cell in the interphase, with rate d_2, and δ the death of a cell in the mitotic phase, with rate d_3. All the rates in the model refer to mass action kinetics.

The BIO-PEPAD model is defined by the following species definitions:

$$T_I \stackrel{def}{=} (\alpha, 1)\downarrow + (\beta, 2)\uparrow + (\gamma, 1)\downarrow$$
$$T_M \stackrel{def}{=} (\alpha, 1)\uparrow + (\beta, 1)\downarrow + (\delta, 1)\downarrow$$

where the species behave as reactants or products, depending on their role as previously specified. Also, as all the actions obey a mass action kinetic law, we simply assume $f_\alpha = f_{MA}(a_1)$, $f_\beta = f_{MA}(a_4)$, $f_\gamma = f_{MA}(d_2)$ and $f_\delta = f_{MA}(d_3)$. The BIO-PEPAD process modeling the interactions is given by

$$T_I(n_0^I) \underset{\{\alpha,\beta\}}{\bowtie} T_M(n_0^M)$$

where n_0^I and n_0^M represent the initial concentration levels for the cells in the interphase and in the mitotic phase, respectively. Notice that γ and δ are not in the cooperation set since model

reactions involving a single species. Also, we recall that this is also a valid Bio-PEPA process specification.

A delay σ' is used to model the duration of the interphase, hence the passage of a tumor cell from the population of those in the interphase to the population of those in the mitotic phase, namely the event modeled by action α, is delayed. To specify the delay in the Bio-PEPAd system to analyze, it is enough to define a function σ where

$$\sigma(\alpha) = \sigma' \qquad\qquad \sigma(\beta) = \sigma(\gamma) = \sigma(\delta) = 0.$$

As a consequence, the Bio-PEPAd process initialized by applying function μ, namely the process configuration $T_I(n_0^I, [\,]) \underset{\{\alpha,\beta\}}{\bowtie} T_M(n_0^M, [\,])$, together with the function σ, completes the definition of the Bio-PEPAd system representing the cell cycle model.

By applying one of the techniques discussed in this paper this system can be analyzed. In particular, the Bio-PEPAd model can be automatically translated into a set of DDEs by applying the algorithm presented in Section 9.2.3. By computing the following vector of the kinetic laws

$$\begin{aligned}\nu_{KL} &= (a_1 T_I(t-\sigma(\alpha)),\ a_4 T_M(t-\sigma(\beta)),\ d_2 T_I(t-\sigma(\gamma)),\ d_3 T_M(t-\sigma(\delta)))^T \\ &= (a_1 T_I(t-\sigma'),\ a_4 T_M(t),\ d_2 T_I(t),\ d_3 T_M(t))^T\end{aligned}$$

the following set of DDEs can be computed:

$$\frac{dT_I}{dt} = 2a_4 T_M - d_2 T_I - a_1 T_I(t-\sigma') \qquad \frac{dT_M}{dt} = a_1 T_I(t-\sigma') - d_3 T_M - a_4 T_M\,.$$

As expected, this DDEs system is analogous to the one presented in Section 3.2 and, as a consequence, by using the same initial conditions it could be analyzed.

As far as the stochastic analysis of the Bio-PEPAd systems is concerned, we can notice that the system we defined corresponds to the following set of reactions

$$T_I \xrightarrow{a_1,\sigma} T_M \qquad\qquad T_M \xrightarrow{a_4} 2T_I$$
$$T_I \xrightarrow{d_2} \epsilon \qquad\qquad T_M \xrightarrow{d_3} .$$

Again, this is exactly the same reactions–based model used as input of both the DSSAs presented in Section 4.2 and 5.2 to compare the deterministic and the stochastic models for the cell cycle.

Chapter 10

Conclusions

In this thesis we focused on stochastic simulation and formal modeling of biological systems with delays. This resulted in a comprehensive study of DSSAs and formal methods.

As regards the former part, we rigorously analyzed five algorithms for simulating stochastic systems with delays, and we proved relations among them. Four of these algorithms are based on some scheduling policy for reactions with delays. We analyzed the DDA, we argued that such an algorithm is not suitable to simulate systems in which species involved in a delayed interaction can be involved at the same time in other interactions. We both experimentally and theoretically validated this conclusion. As far as the experiment is concerned, we compared DDEs and DDA simulations of a model of the cell cycle with delay, observing quantitatively different dynamics. The study of the mathematical foundations of the DDA proved its correctness and associated the algorithm with a DCME describing systems where to each reaction firing correspond two detached state changes. We then defined an algorithm, the PDA, properly behaving in simulating such systems. Again, we experimented the algorithm on the same model used to analyze the DDA, yielding to results which improved DDA simulations. Mathematical foundations of the PDA have been similarly studied, we proved its correctness and we associated the algorithm with a DCME describing systems where to each reaction firing correspond a unique state change, as required. We went through a refinement process for this new algorithm. We started analyzing a naïve version of the PDA potentially inaccurate under some circumstances, so we defined a PDA with markings and we then combined such an algorithm with the DDA.

A last algorithm concludes the first part. Such an algorithm, the DPF, is a DDEs-inspired DSSA where delays represent dependancies on past states of the simulated system. This is in contrast with the other four algorithms, all based on some scheduling policy for firing reactions with delays. Investigating the mathematical foundations of this new algorithm we prove it to be equivalent to the PDA.

In the latter part of the thesis we moved working on actions with delays in process algebras. To this extent, we used as target process algebras CCS and Bio-PEPA. This guarantees to have a quite satisfactory introduction to this topic since all the major process algebras are either CCS-based or CSP-based, as it is the case for Bio-PEPA.

In the case of CCS, we extended CCS to support non-instantaneous actions. In doing that, we concentrated on actions following either the delay-as-duration or the purely delayed approach, yielding to two new algebras CCSD and CCSP, respectively. Both the languages are conservative extensions of CCS, however both languages are able to describe processes in which non-instantaneous actions are started and not completed. This is done by the introduction of a notion of process configuration. Notions of competing actions internal to processes in the case of CCSP are defined, and the effect of non-instantaneous actions on classical CCS operators is discussed. For both the languages we defined a SOS in the ST-style. Moreover, bisimulation relations for both the languages are defined and proved to be congruences.

In the final part of the thesis we enriched the stochastic process algebra Bio-PEPAD with the possibility of assigning delays to actions following the delay-as-duration approach, yielding a new

non-Markovian stochastic process algebra: Bio-PEPAd. As in CCSd and CCSp, Bio-PEPAd. processes are defined with the original syntax of Bio-PEPA. We introduced notions similar to the ones required in CCS with delayed actions: process configurations and systems. We also defined a SOS for Bio-PEPAd. This semantics is again defined in the ST-style since this permits to easily model durational actions. This new algebra allows multiple analysis techniques, in fact we defined the encoding of Bio-PEPAd models in Gsmps, a class of non-Markov processes, in input for the DDa and in sets of DDes. We also proved results stating the relation between Bio-PEPA and Bio-PEPAd models.

Before concluding, we briefly discuss possible future works. Future work can either be planned in the area of DSSAs or in formal methods. As we said, in the non-delayed framework approximations of exact SSAs have been defined. Such algorithms improve substantially computational performance of SSAs and, in some circumstances, the loss of precision due to approximation can be bounded. These approximated algorithms found their mathematical foundations in Stochastic Differential Equations such as the Langevin equation. In the case of DSSAs, the performance of the algorithm is even an harder problem since any of the scheduling-based algorithms we presented is more costly than the SSA. This is obviously due to the scheduling list or history tracking which increase space complexity of simulations. In this sense, Delay Stochastic Differential Equations could be used to define approximated DSSAs.

Moreover, another technique used to improve simulation performances consists of defining hybrid models. Such models are both deterministic and stochastic and, to be simulated, combination of the SSA with both numerical and analytical solution of ODEs must be used (Caravagna *et al.*, 2010). In this sense, it could be investigated whether it is possible to define hybrid models with delays which could be analyzed by properly combining DSSAs and solutions of DDEs.

Furthermore, as we said DDEs are general enough to capture various forms of delays. We restricted our work to consider only constant delays. We argue that the DSSAs we defined could be easily extended to systems where delays are associated to random variables. In this sense, it would be necessary to sample a value for a delay distribution each time a reaction fires. Scheduling policies or history-dependent functions could be extended to this scenario straightforwardly. A similar extension would permit to use variable delays in DSSAs. Of course, approximations and hybrid models could be defined by combining both the forms of delays. In the same fashion, the algorithm using history-dependent propensity functions could be used as a base for a DSSA with distributed delays.

Possible future works in formal methods could similarly be done. Delays, as it is clear, are properties of the reactions in a system. In this sense, delays are kinetic properties of actions in the context of process algebras, and properties of rewriting rules in term rewriting systems. This candidates delays to be a feature that any language, either process-based or term-based, could try to offer. In Bio-PEPA, for instance, delays following the purely delayed approach and combinations of this algebra with Bio-PEPAd should be defined. At the mathematical level of these languages, stochastic semantics with non-instantaneous actions could be defined and relation with non-Markov stochastic processes could be investigated.

We remark that embedding delays in formal methods is a non trivial task. Once that the framework of simulation is extended with more complex forms of delay we would be expected to extend languages to support such new forms of delays. We argue that, even in the case of the intuitive extension to variable delays, this would require a good amount of work.

Finally, once that the theory of DSSAs and languages supporting delayed reactions have been defined, applications should be defined. We argue that most of the exisitng model specifications could be used to easily create models with delays since, in most cases, we are able to retain the original syntax of languages.

Bibliography

van der Aalst, W.M.P. , 1993. Interval Timed Coloured Petri Nets and their Analysis. *Lecture Notes in Computer Science* **691**, 453–472.

Aceto, L., Fokkink, W.J., Verhoef, C., 2001. Structural operational semantics. Chapter in: J.A. Bergstra, A. Ponse and S.A. Smolka editors: Handbook of Process Algebra, 197-292.

Agarwal, R.P., O'Regan, D., 2008. An Introduction to Ordinary Differential Equations. Springer.

Akman, O.E., Ciocchetta, F., Degasperi, A., Guerriero, M.L., 2009. Modelling biological clocks with Bio-PEPA: Stochasticity and robustness for the Neurospora crassa circadian network. Proceedings of CMSB, *Lecture Notes in Computer Science* **5688**, 52-67.

Alur, R., Dill, D.L. , 1994. A Theory of Timed Automata. *Theoretical Computer Science* **126**, 183-235.

Alur, R., Courcoubetis, C., Dill, D. L. 1992. Verifying automata speciÞcations of probabilistic real-time systems. In Proceedings of REX Workshop, *Lecture Notes in Computer Science* **600**, 28-44.

Anderson, D.F., 2007. A modiÞed next reaction method for simulating chemical systems with time dependent propensities and delays. *The Journal of Chemical Physics* **127**, 214107.

Arino, O., Gyori, I., 1989. Necessary and suffcient condition for oscillation of a neutral differential system with several delays. *Journal of Differential Equations* **81**, 98-105.

Arino, O., Gyori, I., Jawhari, A., 1984. Oscillation criteria in delay equations. *Journal of Differential Equations* **53**, 115-123.

Aziz, A., Kanwal, K., Singhal, V., Brayton, V., 1996. Verifying Continuous Time Markov Chains. *Lecture Notes in Computer Science* **1102**, 269-276.

Baeten, J., Bergstra, J., 1991. Real time process algebra. *Formal Aspects of Computing* **3**, 142-188.

Baeten, J., Bergstra, J., 1997. Discrete time process algebra: absolute time, relative time and parametric time. *Fundamenta Informaticae* **29**, 51-76.

Baier, C., Katoen, J.P., Hermanns, H., 1999. Approximate Symbolic Model Checking of Continuous-Time Markov Chains. *Lecture Notes in Computer Science* **1664**, 146-161.

Baker, C.T.H., Bocharov, G.A., Rihan, F.A., 1999. A Report on the Use of Delay Differential Equations in Numerical Modelling in the Biosciences. Technical Report, The University of Manchester, UK.

Banatre, J.P., Fradet, P., Le Métayer, D., 1996. Gamma and the chemical reaction model: ten years after. *Coordination programming*, 3-41.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., 2009(a). An Intermediate Language for the Stochastic Simulation of Biological Systems. *Theoretical Computer Science* **410**, 3085-3109.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., 2009(b). On the Interpretation of Delays in Delay Stochastic Simulation of Biological Systems. Proceedings of CompMod'09, *Electronic Proceedings in Theoretical Computer Science* **6**, 17-29.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., 2011(a). Delay Stochastic Simulation of Biological Systems: A Purely Delayed Approach. C. Priami et al. (Eds.), *Transactions on Computational Systems Biology* XIII, LNBI **6575**, 61-84.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., 2011(b). A Delay Stochastic Simulation Algorithm with Delayed Propensity Functions. In preparation.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., 2008(a). The Calculus of Looping Sequences. In M.Bernardo, P.Degano and G.Zavattaro editors: Formal Methods for Computational Systems Biology, *Lecture Notes in Computer Science* **5016**, 387-423.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., Tini, S., 2010(b). Aspects of multiscale modelling in a process algebra for biological system Proceedings of MeCBIC 2010, *Electronic Proceedings in Theoretical Computer Science* **40** , 54-69.

Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., Tini, S., 2011(b). Foundational aspects of multiscale modelling of biological system with process algebras. Submitted to *Theoretical Computer Science*.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., 2008(b). Spatial Calculus of Looping Sequences. *Electronic Notes in Theoretical Computer Science* **229** (1), 21-39.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tini, S. 2008(c). Compositional semantics and behavioral equivalences for P Systems. *Theoretical Computer Science* **395**(1), 77-100.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tini, S. 2008(d). A P Systems Flat Form Preserving Step-by-step Behaviour. *Fundamenta Informaticae* **87**(1), 1-34.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tini, S. 2010(c). Compositional semantics of spiking neural P systems. *Journal of Logic and Algebraic Programming* **79**(6), 304-316.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tini, S. 2010(d). An Overview on Operational Semantics in Membrane Computing. *International Journal of Foundations of Computer Science* **22**(1), 119-131.

Barrio, M., Burrage, K., Leier, A., Tian Y., 2006. Oscillatory Regulation of Hes1: Discrete Stochastic Delay Modelling and Simulation. *PLoS Computational Biology*, **2** (9), e117.

Beretta, E. Hara, T., Ma, W., Takeuchi, Y., 2002. Permanence of an SIR Epidemic Model with Distributed Time Delays. *Tohoku Mathematical Journal* **54** (2), 581-591.

Bratsun, D.,ÊVolfson, D., Tsimring, L.S., Hasty, J., 2005. Delay-induced Stochastic Oscillations in Gene Regulation. *Proceedings of the National Academy of Sciences* **102** (41), 14593-14598.

Bravetti, M., Gorrieri, R., 1999. Deciding and Axiomatizing ST Bisimulations for a Process Algebra with Recursion and Action ReÞnement. Technical Report UBLCS-99-1, University of Bologna, Italy.

Bravetti, M., Gorrieri, R., 2002. The theory of interactive generalized semi-Markov processes. *Theoretical Computer Science* **282** (1) , 5-32.

Bravetti, M., Bernardo, M., Gorrieri, R., 1998. Towards Performance Evaluation with General Distributions in Process Algebras. Proceedings of CONCUR Õ98, *Lecture Notes in Computer Science* **1466** (1) , 405-422.

Tian, T., Burrage, K., Burrage, P.M., Carletti, M., 2007. Stochastic delay differential equations for genetic regulatory networks. *Journal of Computational and Applied Mathematics* **205**, 696-707.

Cai, X., 2007. Exact stochastic simulation of coupled chemical reactions with delays. *The Journal of Chemical Physics* **126**, 124108.

Calder, M., Gilmore, S., Hillston, J., 2006. Modelling the inSSuence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Transactions on Computational Systems Biology* **VII** 4230, 1-23.

Cao, Y.,Gillespie, D.T., Petzold, L.R., 2005. The Slow-scale Stochastic Simulation Algorithm. *Journal of Chemical Physics* **122** (1), 014116.

Caravagna, G., D'Onofrio, A., Milazzo, P., Barbuti, R., 2010. Antitumour Immune Surveillance Through Stochastic Oscillations. *Journal of Theoretical Biology* **265** (3), 336-345.

Caravagna, G., Hillston, J., 2010. Modeling biological systems with delays in Bio-PEPA. Proceedings of MeCBIC 2010, *Electronic Proceedings in Theoretical Computer Science* **40**, 85-101.

Caravagna, G., Hillston, J., 2011. Bio-PEPAd: a non-Markovian extension of Bio-PEPA. Submitted to *Theoretical Computer Science*.

Cardelli, L., 2005. Brane calculi. In Vincent Danos and Vincent Schachter, editors, Proceedings of CMSB, *Lecture Notes in Computer Science* **3082**, 257-280.

Cardelli, L., Păun, G. 2006. An universality result for a (mem)brane calculus based on mate/drip operations. *International Journal of Foundations of Computer Science* **17**(1), 49-68.

Cardelli, L., Caron, E., Gardner, P., Kahramanogullari, O., Phillips, A., 2009. A process model of Rho GTP-binding proteins. *Theoretical Computer Science* **410** (33-34), 3166-3185.

Cassandras, C.G., Lafortune, S., 2007. Stochastic Timed Automata. Chapter In: Introduction to Discrete Event Systems, Springer.

Chen, K.C., Calzone, L., Csikasz-Nagy, A., Cross, F.R. Novak, B., Tyson, J.J., 2004. Integrative analysis of cell cycle in budding yeast. *Molecular Biology of the Cell* **15**, 3841-3862.

Ciocchetta, F., Hillston, J., 2009. Bio-PEPA: a Framework for the Modelling and Analysis of Biochemical Networks. *Theoretical Computer Science* **410** (33-34), 3065-3084.

Ciocchetta, F., Hillston, J., 2008. Calculi for Biological Systems. Chapter in: M.Bernardo, P.Degano and G.Zavattaro (Eds.): Formal Methods for Computational Systems Biology (SFM 2008), *Lecture Notes in Computer Science* **5016**, 265-312.

Corradini, F., 2000. Absolute versus Relative Time in Process Algebras. *Information and Computation* **156** (1-2), 122-172.

Cox, D.R., 1955. The Analysis of non-Markovian Stochastic Processes by the Inclusion of Supplementary Variables. *Proceedings of Cambridge Philosophical Society* **51**, 433-440.

Csikasz-Nagy, A., Battogtokh, D., Chen, K.C., Novak, B., Tyson, J.J., 2006. Analysis of generic model of eukaryotic cell-cycle regulation. *Biophysical Journal* **90**, 4361-4379.

D'Onofrio, A., Manfredi, P., Salinelli, E., 2007. Vaccinating behaviour, information, and the dynamics of SIR vaccine preventable diseases. *Theoretical Population Biology* **71**, 301-317.

Damgaard, T.C., Danos, V., Krivine, J., 2008(a). A language for the cell. Technical Report TR-2008-116, IT University of Copenhagen, Denmark.

Damgaard, T.C., Danos, V., Krivine, J., 2008(b). A language for the cell. Technical Report TR-2008-116, IT University of Copenhagen, 2008. A generic language for biological systems based on Bigraphs. Technical Report TR-2008-115, IT University of Copenhagen, Denmark.

Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J., 2009(a). Rule-based modelling and model perturbation. *Transactions on Computational Systems Biology* **5750** (11), 116-137.

Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J., 2009(b). Rule-based modelling of cellular signalling. In L. Caires and V. T. Vasconcelos, editors, Proceedings of CONCUR, *Lecture Notes in Computer Science* **4703** 17-41. Tutorial paper.

De Simone, R.,1984. Calculabilité et Expressivité dans l'Algebre de Processus Paralleles MEIJE. Theses de 3^e cycle, Universitee Paris VII, France.

De Simone, R.,1985. Higher-level synchronizing devices in MEIJE-CCS. *Theoretical Computer Science* **30**, 133-138.

Dematté, L.,Priami, C., Romanel, A., 2008. Modelling and simulation of biological processes in BlenX. *SIGMETRICS Performance Evaluation Review* **35** (4), 32-39.

Driver, R.D., 1977. Ordinary and Delay Differential Equations. Springer.

Driver, R.D., 1962. Existence and stability of solutions of a delay-differential system. *Rational Mechanical Analysis* **10**, 401-426.

Escalante, M.A., Dimopoulos, N.J., 1994. A Probabilistic Approach to Timing Analysis for Synthesis. Technical Report ECE-94-6. University of Victoria, Canada.

Galpin, V., Hillston, J., 2011. A semantic equivalence for Bio-PEPA based on discretisation of continuous values. To appear in *Theoretical Computer Science*.

Gibson, M.A., Bruck, J., 2000. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *Journal of Physical Chemistry A* **104** (9), 1876-1889.

Gillespie, D.T., 1976. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics* **22** (4) 403-434.

Gillespie, D.T., 1977. Exact Stochastic Simulation of Coupled Chemical Reactions. *Journal of Physical Chemistry* **81** 2340-2361.

Gillespie, D.T., 2001. Approximated Accelerated Stochastic Simulation of Chemically Reacting Systems. *Journal of Chemical Physics* **115** (4), 1716-1733.

Gillespie, D.T., Petzold, L.R., 2006. Numerical Simulation for Biochemical Kinetics. Chapter in: System modeling in cell biology : from concepts to nuts and bolts, MIT Press.

van Glabbeek, R.J., 1990(a). The linear time – branching time spectrum I (the semantics of concrete, sequential processes). *Lecture Notes in Computer Science* **458**, 278-297.

van Glabbeek, R.J., 1990(b). The ReÞnement Theorem for ST-Bisimulation Semantics. Proceedings of the IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET '90) (Sea of Gallilea, Israel), 27-52.

van Glabbeek, R.J., 1993. The linear time – branching time spectrum II (the semantics of sequential systems with silent moves). *Lecture Notes in Computer Science* **715**, 66-81.

van Glabbeek, R.J., Vaandrager, F.W., 1987. Petri Net Models for Algebraic Theories of Concurrency. *Lecture Notes in Computer Science* **259**, 224-242.

Glynn, P.W., 1983. On the Role of Generalized Semi-Markov Processes in Simulation Output Analysis. Proceedings of the 15th conference on Winter simulation - Volume 1, 39-44.

Guerriero, M.L., Heath, J.K., Priami, C., 2007. An automated translation from a narrative language for biological modelling into process algebra. In M. Calder and S. Gilmore, editors, Proceedings of CMSB, *Lecture Notes in Computer Science* **4695**, 136-151.

Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O., 2008. Probabilistic Model Checking of Complex Biological Pathways. *Theoretical Computer Science*, Special Issue on Converging Sciences: Informatics and Biology, **391**, 239- 257.

Hennessy, M., 1988. Axiomatising Finite Concurrent Processes. *SIAM Journal of Computing* **17** (5), 997-1017.

Hennessy, M., Regan, T., 1988. A process algebra for timed systems. *Information and Computation* **177**, 221Ð239.

Hillston, J., 1996. A Compositional Approach to Performance Modelling. Cambridge University Press.

Ideker, T., Galitski, T., Hood, L., 2001. A new approach to decoding life: Systems biology. *Annual Review of Genomics and Human Genetics* **2**, 343-372.

Jansen, A.R.J., 1995. Monte Carlo simulations of chemical reactions on a surface with time-dependent reaction-rate constants. *Computer Physics Communications* **86**, 1-12.

Jensen, K., 1992. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Monographs in Theoretical Computer Science, Springer.

Kermack, W.O., McKendrick, A.G., 1927. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London* **115** A, 700-721.

Kwiatkowska, M., Norman, G., Parker, D., 2007. Stochastic model checking. *Lecture Notes in Computer Science* **4486**, 220-270.

Kwiatkowski, M., Stark, I., 2008. The continuous π-calculus: a process algebra for biochemical modelling. In M. Heiner and A. M. Uhrmacher, editors, Proceedings of CMSB, *Lecture Notes in Computer Science*, **5307**, 103-122.

Kitano, H., 2001. Systems biology: Towards system-level understanding of biological systems. *Foundations of Systems Biology*, 1-36.

Kouyous, R.D., Althaus, C.L., Bonhoeffer, S., 2006. Stochastic or deterministic: what is the effective population size of HIV-1? *Trends in Microbiology* **14**, 507-511.

Krivine, J., Milner, R., Troina, A., 2008. Stochastic Bigraphs. *Electronic Notes in Theoretical Computer Science* **218**, 73-96.

Le Novére, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J.L., Hucka, M., 2006. BioModels Database: a Free, Centralized Database of Curated, Published, Quantitative Kinetic Models of Biochemical and Cellular Systems. *Nucleic Acids Research* **34** D689-D69.

Leea, D.Y., Zimmerc, R., Leea, S.Y., Park, S., 2006. Colored Petri net modeling and simulation of signal transduction pathways. *Metabolic Engineering* **8** (2) 112-22.

Leier, A., Burrage, K., Burrage, P., 2007. Stochastic Modelling and Simulation of Coupled Autoregulated Oscillators in a Multicellular Environment: The her1/her7 Genes. *Lecture Notes in Computer Science* **4487** , 778-785.

Li, F., Long, T., Lu, Y., Ouyang, Q., Tang, C., 2004. The yeast cell cycle is robustly designed. *Proceedings of the National Academy of Sciences* **101** (14), 4781-4786.

Lotka, A.J., 1920. Undamped oscillations derived from the law of mass action. *Journal of the American Chemical Society* **42**, 1595.

Marquez-Lago, T.T., Leier, A., Burrage, K., 2010. Probability distributed time delays: integrating spatial effects into temporal models. *BMC Systems Biology* **4**:19.

Marsan, M.A., 1989. Stochastic Petri Nets: an Elementary Introduction. In G.Rozenberg (Editor), Advances in Petri Nets, *Lecture Notes in Computer Science* **424**.

Martin, A., Ruan, S., 20001. Predator-prey Models with Delay and Prey Harvesting. *Journal of Mathematical Biology* **43** (3), 247Ð267.

Matthes, K., 1962. Zur Theorie der Bedienungsprozesse. In Transactions of the 3rd Prague Conference on Information Theory and Statistical Decision Functions, 513-528.

Milazzo, P., 2007. Qualitative and Quantitative Formal Modeling of Biological Systems. Ph.D. Thesis, Department of Computer Science, University of Pisa, Italy.

Milner, R., 1980. A Calculus of Communicating Systems. *Lecture Notes in Computer Science* **92**.

Milner, R., Parrow, J., Walker, D., 1992. A calculus of mobile processes *Information and Computation* **100**, 1-40.

Milner, R., Tofte, M., Harper, R., 1990. The deÞnition of Standard ML. MIT Press.

Moller, F., Tofts, C., 1990. A temporal calculus of communicating systems. In Baeten, J., Klop, J., eds.: Proceedings of CONCUR, *Lecture Notes in Computer Science* **458**, 401-415.

Momiji, H., Monk, N.A.M., 2008. Dissecting the dynamics of the Hes1 genetic oscillator. *Journal of Theoretical Biology* **254**, 784-798.

Monk, N.A.M., 2003. Oscillatory Expression of Hes1, p53, and NF-κB Driven by Transcriptional Time Delays. *Current Biology* **13**, 1409-1413.

Murata, T., 1989. Petri nets: properties, analysis and applications. *Proceedings of IEEE*, **77** (4), 541-580.

Murray, J., 1989. Mathematical Biology. Springer.

Nicollin, X., Sifakis, J., 1994. The algebra of timed processes, ATP: Theory and application. *Information and Computation* **114**, 131-178.

Palamidessi, C., 2003. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. *Mathematical Structures in Computer Science*, **13**(5), 685–719.

Păun, G., 2002. Membrane Computing. An Introduction. Springer.

Păun, G., Rozenberg, G., Salomaa, A., 1998. DNA Computing - New Computing Paradigms. Springer.

Park, D.M.R., 1981. Concurrency and automata on inÞnite sequences. *Lecture Notes in Computer Science* **104**, 167-183.

Pedersen, M.D., Plotkin, G., 2010. A Language for Biochemical Systems: Design and Formal Specification. *Theoretical Computational Systems Biology* **12**, 77-145.

Plotkin, G., 1981. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Aarhus University, Denmark.

Plotkin, G., 2004. The Origins of Structural Operational Semantics. *Journal of Logic and Algebraic Programming* **60-61**, 3-15.

Priami, C., 1995. Stochastic pi-calculus. *The Computer Journal* **38** (7), 578-589.

Priami, C., Quaglia, P., 2005. Beta binders for biological interactions. In V. Danos and V. Schachter, Proceedings of CMSB *Lecture Notes in Bioinformatics* **3082**, 20-33.

Ramchandani, C., 1973. Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets. PhD thesis, Massachusetts Institute of Technology, USA.

Rathinam, M., Petzold, L.R., Cao, Y., Gillespie, D., 2003. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics* **119** (24), 12784-12794.

Reddy, V.N., Mavrovouniotis, M.L., Liebmant, M.N., 1993. Petri net representations in metabolic pathways. Proceedings of the International Conference on Intelligent Systems for Molecular Biology, 328-336.

Regev, A., Paninab, E.M., Silverman, W., Cardelli, L., Shapiro, E., 2004. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science* **325** (1), 141-167.

Regev, A., Silverman, W., Shapiro, E., 2001. Representation and simulation of biochemical processes using the pi-calculus process algebra. In PaciÞc Symposium on Biocomputing, 459-470.

Reisig, W., 1985. Petri nets: an Introduction. Prentice–Hall.

Ribeiro, A.S. Smolander, O.P., Rajala, T., Hakkinen, A., Yli-Harja, O., 2009. Delayed Stochastic Model of Transcription at the Single Nucleotide Level. *Journal of Computational Biology* **16** (4), 539-553.

Ross, S.M., 1995. Stochastic Processes. Wiley.

Ruan, S., 2009. On Nonlinear Dynamics of Predator-Prey Models with Discrete Delay. *Mathematical Modeling Natural Phenomena* **4** (2), 140-188.

Roussel, M.R., Zhu, R., 2006. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Physical Biology* **3**, 274-284.

Sauer, U., Heinemann, M., Zamboni, N., 2007. Getting Closer to the Whole Picture. *Science* **316**, 550-551.

Schlicht, R., Winkler, G., 2008. A delay stochastic process with applications in molecular biology. *Journal of Mathematical Biology* **57**, 613-648.

Segel, I.H., 1993. Enzyme Kinetics: Behaviour and Analysis of Rapid Equilibrium and Steady-State Enzyme Systems. Wiley.

Shahrezaei, V., Ollivier, J.F., Swain, P., 2008. Colored Extrinsic Fluctuations and Stochastic Gene Expression. *Molecular System Biology* **196** (4).

Sifakis, J. 1977. Use of Petri Nets for Performance Evaluation. *Acta Cybernetica* **4** (2), 185-202.

Smith, H., 2011. An Introduction to Delay Differential Equations with Applications to the Life Sciences. Springer.

Tyson, J.J., 1973. Some further studies of nonlinear oscillations in chemical systems. *Journal Chemical Physics* **58**, 3919-3930.

Vaandrager, F., 1993. Expressiveness results for process algebras. *Lecture Notes in Computer Science* **666**, 609-638.

Villasana, M., Radunskaya, A., 2003. A Delay Differential Equation Model for Tumor Growth. *Journal Mathematical Biology* **47**, 270-294.

Volterra, V., 1926. Fluctuations in the abundance of a species considered mathematically. *Nature* **118**, 558-560.

Wilkinson, D.J., 2006. Stochastic Modelling for Systems Biology. Chapman & Hall/CRC.

Zhanga, F., Lia, Z., Zhangc, F., 2008. Global Stability of an SIR Epidemic Model with Constant Infectious Period. *Applied Mathematics and Computation* **199** (1), 285-291.

Zhou, W., Peng, X., Yan, Z., Wang, Y., 2008. Accelerated stochastic simulation algorithm for coupled chemical reactions with delays. *Computational Biology and Chemistry* **32**, 240-242.

Zhu, R., Ribeiro, A.S., Salahub, D., Kauffman, S.A., 2007. Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models. *Journal of Theoretical Biology* **246** (4) 725-745.

Zuberek, W.M., 1980. Timed Petri Nets and Preliminary Performance Evaluation. Proceedings of the 7th annual Symposium on Computer Architecture, vol. **8** (3) of *Quarterly Publication of ACM Special Interest Group on Computer Architecture*, 62-82.